Estimate Web App

Group H

Introduction

Group introduction

Project overview

- The project was to continue development on an existing web app
- Existing app was buggy & had incomplete implementation of requirements
- Web app intended to aid users in recording and improving their estimation abilities through estimation practice exercises & actual task logging

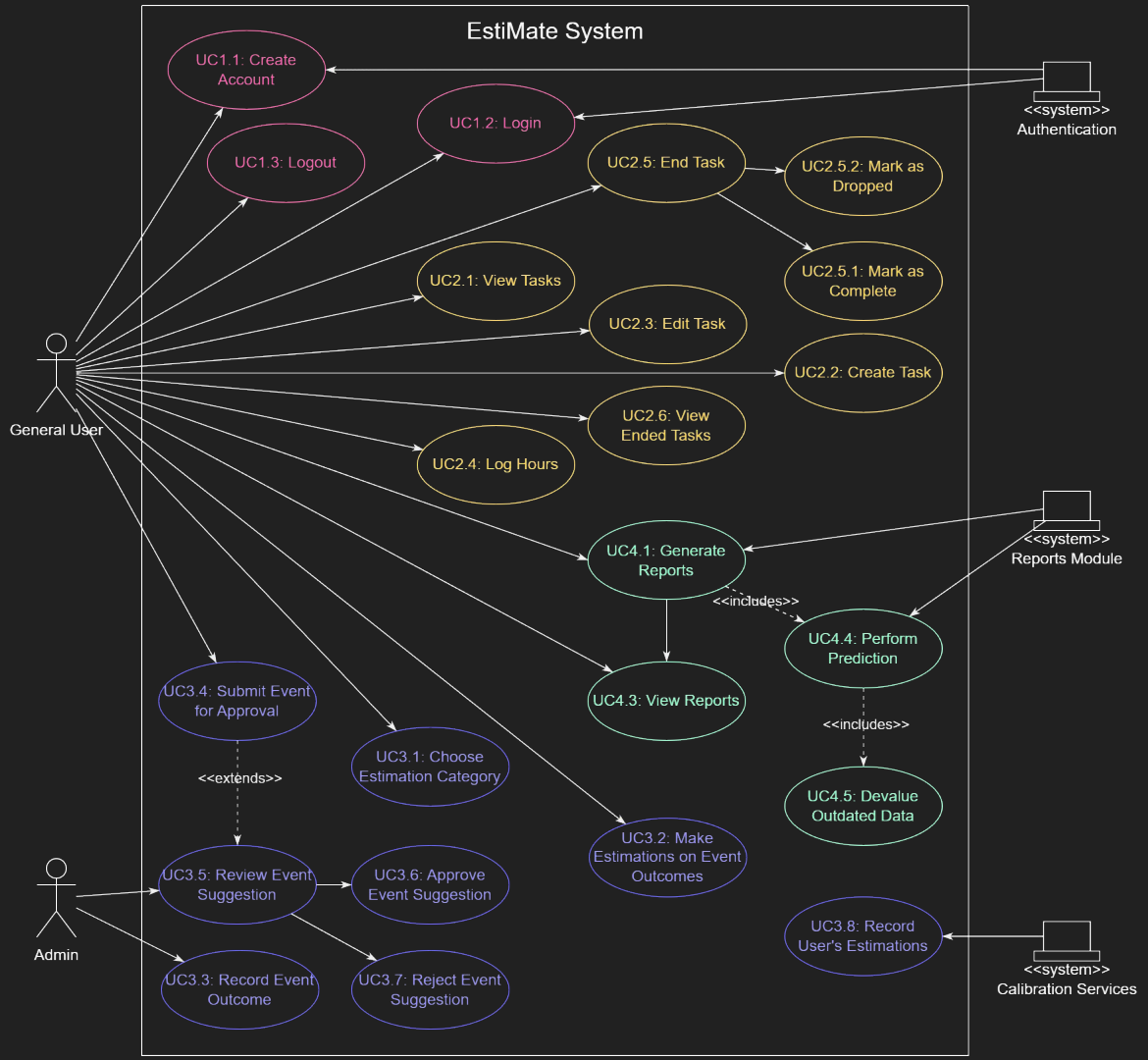
Use Case Diagram

Account

Tasks

Estimation Calibration

Reports



UC-2.6

Name	View Ended Tasks
Requirement(s)	2.5.3
Use Case #	2.6
Actors	User
Trigger	User clicks “View Ended Tasks”
Pre-Conditions	User is logged in and has ended a task (otherwise page will be blank)
Post-Conditions	User is brought to the Ended Tasks page
Basic Flow	1. User clicks “Tasks” in nav bar 2. System displays tasks page 3. User clicks “View Ended Tasks” 4. System displays Ended Tasks page which contains Completed and Dropped tasks pulled from DB

UC-4.3

Name	View Report
Requirement(s)	4.1.2, 4.2.1, 4.3, 4.4
Use Case #	4.3
Actors	User, System
Trigger	User clicks “Reports”
Pre-Conditions	The user has clicked “Generate Report” before so there are reports in the database
Post-Conditions	User is taken to the View Report page
Basic Flow	1. User clicks “Reports” in nav bar 2. System displays reports page 3. User clicks “View Report” on a report in the list 4. System queries database & retrieves report data 5. System displays full report

Added Requirements - Implemented

2.2.4. A new task shall contain an estimation time in hours

2.4.2. The system shall take current date and time when being added

2.4.3. The user shall be able to quickly add hours in increments with buttons

2.4.4. The user shall be able to add custom hours to a task

2.5.3. The user shall be able to view ended tasks

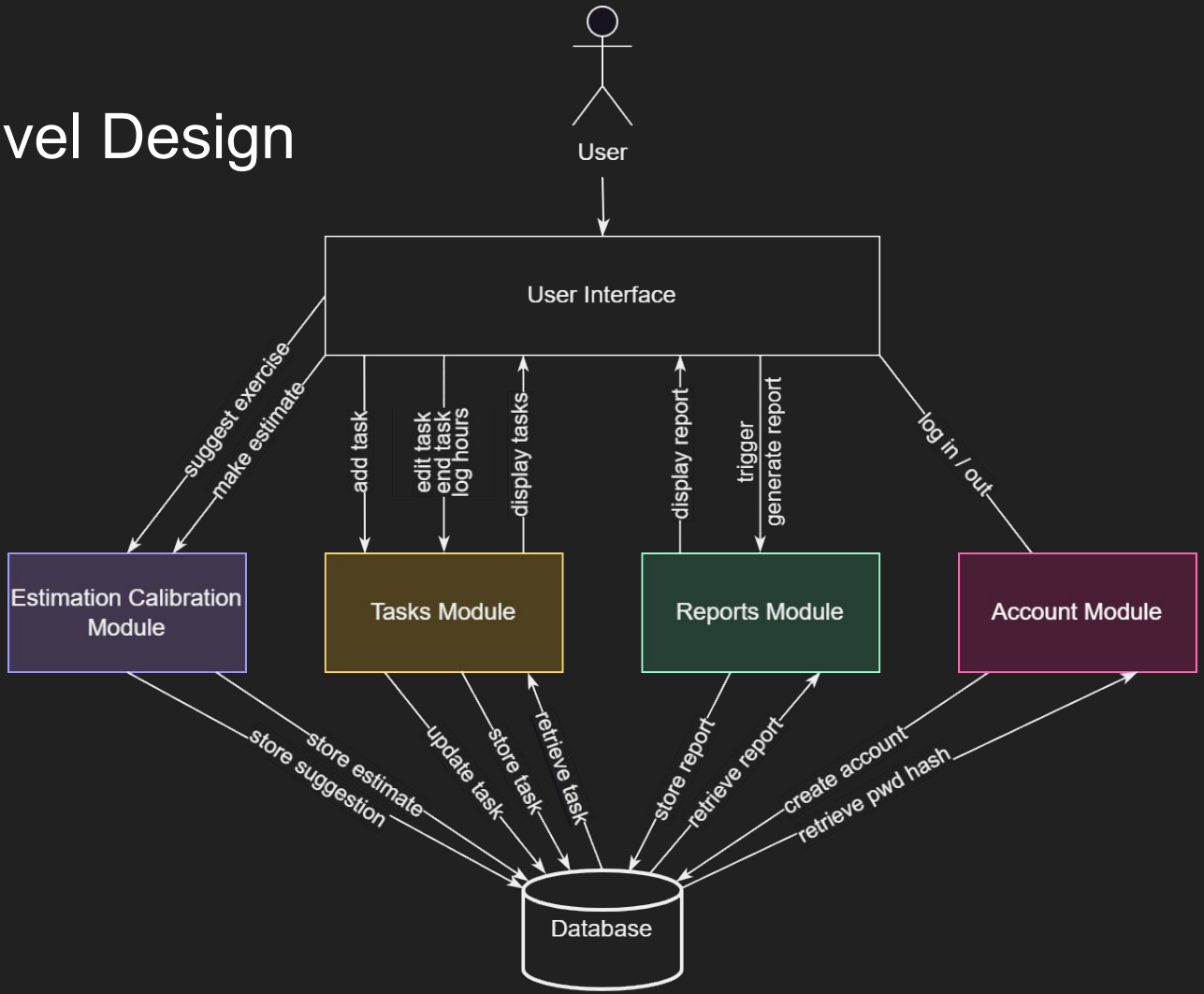
4.2.1. The default report start & end dates shall be generation date of previous report (start) & current timestamp (end)

4.4. Reports shall contain information on the user's activity

4.4.1. Reports shall be generated on the user's logged task hours in relation to the date they occurred

All date/time data shall be stored as TIMESTAMP & retrieved as LocalDateTime

High-Level Design



General Design

Tapestry follows Model-View-Controller pattern:

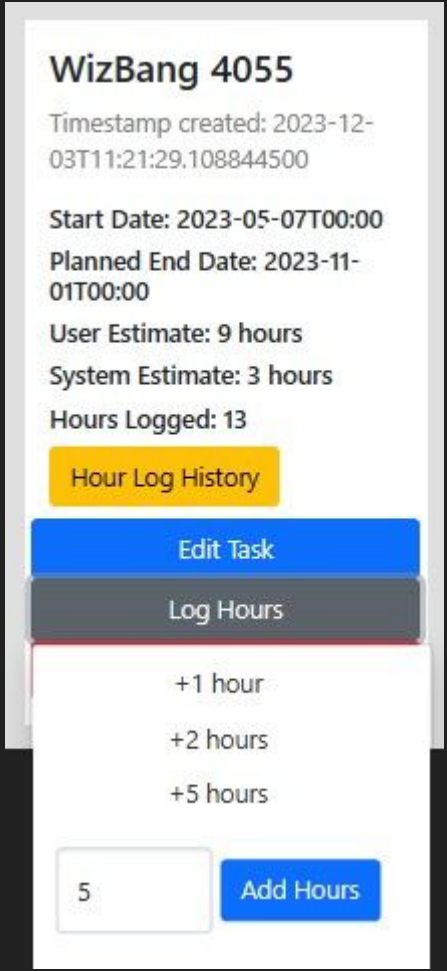
Model	database, designed in Cayenne modeler, Java interfaces for entity types, services query database to retrieve data
View	CSS & .tml files
Controller	Java page classes retrieve data from services and inject into pages

Detailed Design - Add Hours UI

Original UI design only had a text box

Our design adds +1 hour, +2 hours, +5 hours, & custom hour increment buttons

Improves user experience & ease of use

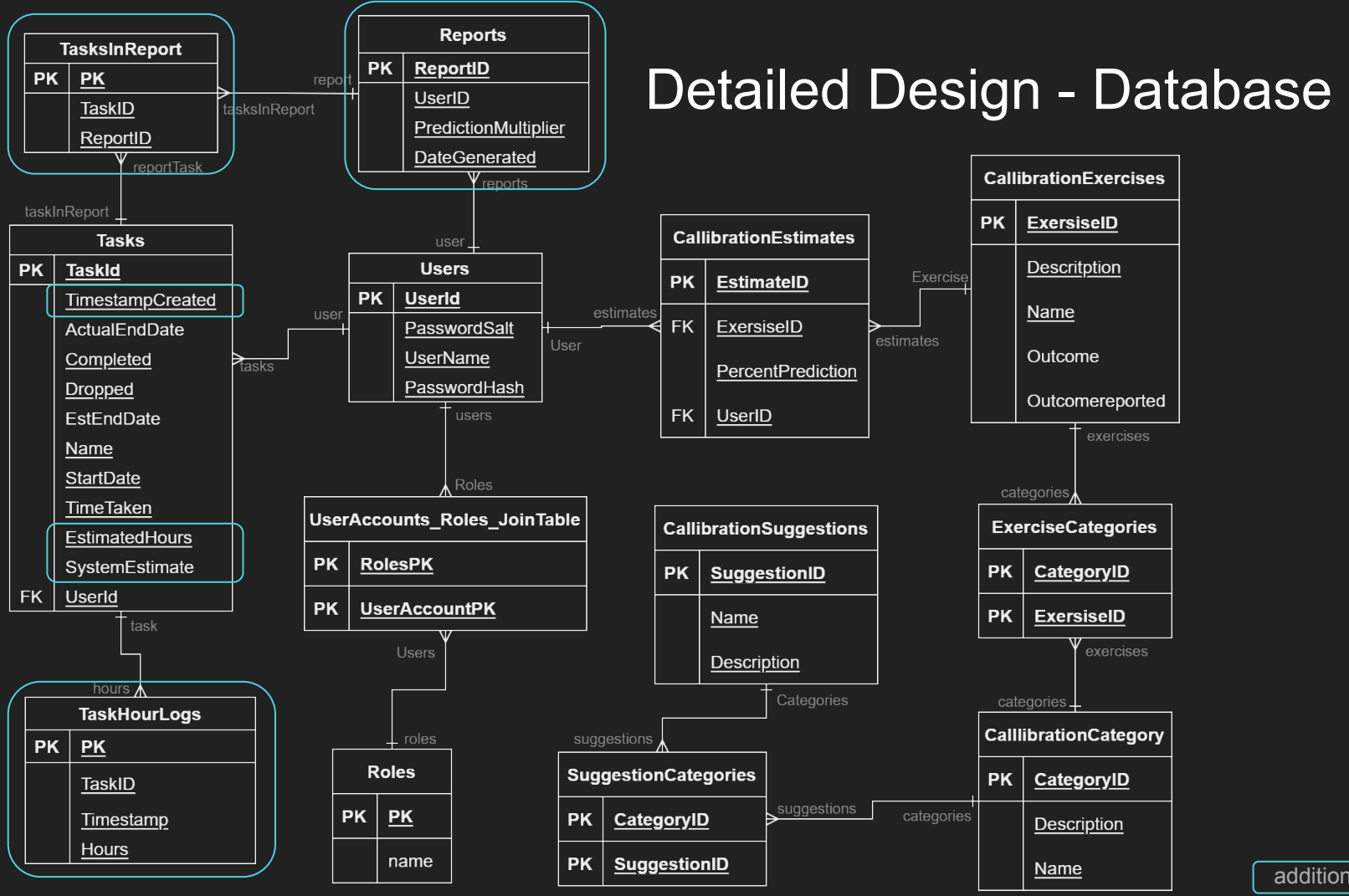


Improved UI

Multiple css improvements

- Tasks now have box shadow to look raised
- Landing page has a slogan and attention grabbing box with hidden content visible on hover
- Whole app is now responsive to multiple views

Detailed Design - Database



Prediction Algorithm

Prediction Multiplier - average actual-to-prediction ratio, default 1.0

- Stored in Reports table
- Devalues old data - weight of **0.1** for data 2+ years old, **0.4** for data 1 - 2 years old, **0.7** for data 0.5 - 1 year old, and **1.0** for data within the last 6 months
- Updated when report created
- Prediction Multiplier = sum of ([actual time taken] ÷ [user estimate] * weight) for all completed tasks

SystemEstimate for tasks = [user estimate] * [most recent prediction multiplier]

Bug Fixes

- Fixed logging hours implementation to prevent negative hours from being logged
- Fixed crash that would happen after the user tries to create more than one task
- Fixed crash that would happen when the user enters invalid input
    - e.g entering text into input field for logging custom number of hours

Alternatives - Unimplemented Requirements

3.3.1. Suggestion shall include date by which the result of the event will be known

3.5. Probability calibration shall contain an estimation slider that begins at 50%

3.5.1. The slider shall increment by 10%

3.6. Probability calibration shall contain a toggle button to indicate whether the event will happen or not

4.1. Reports shall be automatically generated every month

4.1.1. Users should be able to set when reports are generated

4.2. User should have option to generate report between two dates

4.4.2. Reports shall be generated on the user's estimations

5.1.1. Admin shall be able to make edits to suggested exercises for language & readability

Demonstration

Conclusion

- Overall the project turned out better than it was originally, however it could have been better.
- One major issue our group faced was insufficient requirements and documentation. Gaps in requirements were discovered during implementation
  - e.g. were the reports supposed to include data from the calibration exercises?
  - e.g. what is the prediction algorithm supposed to look like?
- We implemented most of the requirements we identified at the beginning
- Lesson learned: implementation stage is easier if you have thoroughly ironed out the requirements
- Future work
  - Straighten out inheritance - e.g. TaskInterface was implemented by many classes that mostly did not need it
  - Implementation of unfinished requirements