Team Number: 7

Team Members:

- Cole
- Blake
- Delaney
- Chris
- Michael

Project Name: SmartScheduler

Project Synopsis:
Web app for students to select courses and visualize their weekly timetable to manage their semester schedule effectively.

Architecture Description:

Tech Stack

Frontend:

- Typescript
- React
- Vite

Backend:

- Node.js

State Management:

- Redux

Sprint 1

Scope:

File Structure Created

Searching:

- User can type in a search box and see class suggestions in a dropdown

Dropdown:

- After the user searches display the results in a dropdown (fake data at this point)

Selection:

- Selecting a class from the dropdown appends it to the selected list

File Structure Breakdown:

The core application directory is “SmartScheduler” which contains configuration files and the following directories: “public” which contains the Vite logo in vite.svg, “server” which has the backend entry point with index.js, and “src” which is the main application source.

Configuration files:

- package.json / package-lock.json: define dependencies and scripts
- vite.config.ts: configuration for the Vite build tool
- tsconfig*.json: TypeScript compiler settings
- eslint.config.js: defines linting rules for code style and quality
- index.html: root HTML entry for the web app

src:

Contains the key files App.tsx, main.tsx, and types.ts as well the following directories: assets, components, features, redux, styles, and utils

- App.tsx: main application component
- main.tsx: app bootstrap file
- types.ts: shared TypeScript types

assets:

- Holds images, SVGs, and other media

components:

- ClassCard.tsx: displays individual class info in left column

- Grid.tsx: layout for schedule visualization

- Header.tsx: top navigation or app banner

- ScheduleCard.tsx, UnscheduledCard.tsx: cards for scheduled and unscheduled classes

- SearchBar.tsx: course search input

- SelectedCourses.tsx, TotalCreditHours.tsx: displays user-selected courses and totals

- UnscheduledTable.ts: classes with no scheduled time

features:

- scheduleSlice.ts: a Redux slice for managing schedule-related state

redux:

- store.ts: configures the Redux store
- hooks.ts: provides typed hooks for Redux operations

styles:

- App.css, HeaderStyles.css, SearchBarStyles.css: scoped styling for UI consistency

utils:

- getCourses.ts: helper for fetching or filtering course data

Components:

Search and Selection Loop:

-Handle events: The user types a course name into the SearchBar component. Input events trigger calls to the SearchServer, which gets course data.

-Update state: The returned course data is dispatched through Redux Actions, updating the Redux State store with new search results and selected course data.

Search:

SearchBar:

- Manages search input, dropdown visibility, and user interactions
- Updates Redux state with the results through dispatched actions
- Handles UI logic for showing/hiding dropdowns

SearchServer:

-Backend proxy between frontend and university

Selection and State Management:

ReduxAction:

- Manages global state changes triggered by the SearchBars
- Handles updating selected class lists.

Redux State:

- Stores all selected courses
- Supplies state context to components like SelectedClasses via Redux hooks

Sprint 2

Scope:

Class cards:

-Class cards will render with fake temporary data

-Class cards will render with real data from API

Credit Hours:

-The total will update and display as classes are added/removed (removing is not Implemented yet.)

Schedule:

-The UI of the schedule grid will be implemented
-Schedule cards will be displayed (fake temp data)
-Unscheduled classes display

Permutations

-Scheduling algorithm is implemented
-Pinning and blocking prepared for next print

File Structure Breakdown

Same as sprint 1

Structure components:

ClassCard.tsx: Update class cards with selected card data

ScheduleCard.tsx: render section times and instructor

ClassGrid.tsx: Display blocked time slots

PermutationSelector.tsx: Users view and cycle through permutations of schedules

TotalCreditHours.tsx: Display updates to credit hour from selected courses.

Features:

ScheduleSlice.ts: Manage new state values blocked times, increment through permutations.

Utils:

Utilities.ts: Added functions getScheduled section, getUnscheduledSections, getPermutations.

getCourse.ts: Parse data from SearchServer

Server:

SearchServer.ts: connect to API, functions getSearchMessage() and get()

Components:

Data Flow:

Search Input - The Search bar sends term to SearchServer

Data Parsing - data is parsed through utilities.parseHTMLResponse into ClassData and SectionData, and updates ScheduleState.

Rendering:
ClassCard display course data

ClassGrid, GridBox render times based on ScheduleTime

PermutationsSelector users scroll through permutations

TotalCreditHours updates when users remove or add classes

Scheduling Algorithm:
Computes all valid permutations of classes that do not overlap and stores them in the Redux state. Current view stored by currentPermutation and change with incramentPermutation and decramentPermutation

Sprint 3

Scope:

Class Cards:

Users can remove class cards from selected classes
Removing updates schedule view

Schedule:

Connecting pinning and blocking to UI

Components:

ScheduleCourses.tsx: update remove for class cards, removes from the selectedCourses redux.and re- renders(Using classGrid and Total credit hours)

ClassGrid.tsx: Visually display blocked and pinned states

ScheduleCard.tsx: render for pinned schedules

Header.tsx update controls to include pinning

Features:

scheduleSlice.ts:

Added remove course, pinSchedule, and blockTime
Implement logic to include pinning in permutations

Utils:
getPermutations.ts: Update for blocked and pinned schedules

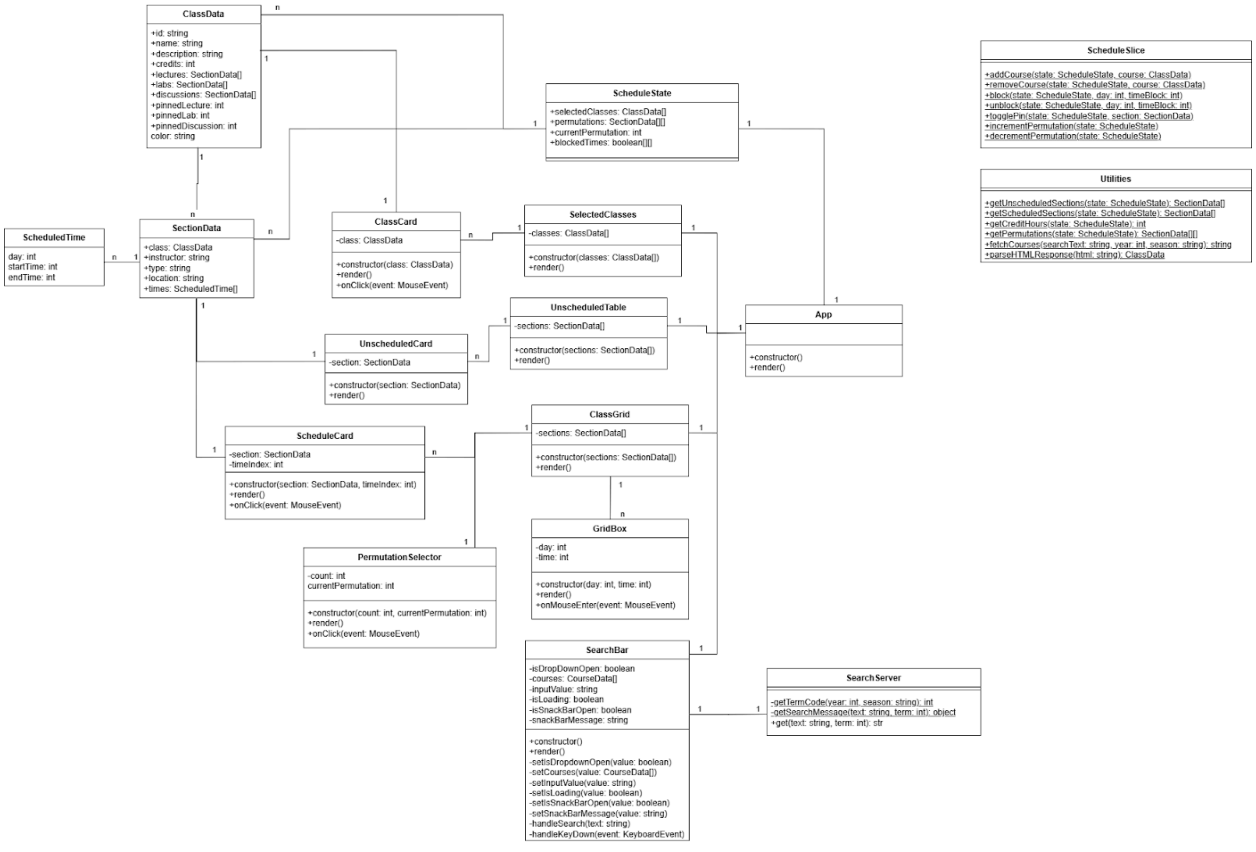
Components:

Removing:
When a user removes a selected course, the course is deleted from the state using the removeCourse. Re render, ClassGrid, TotalCreditHours, and PermutationSelector

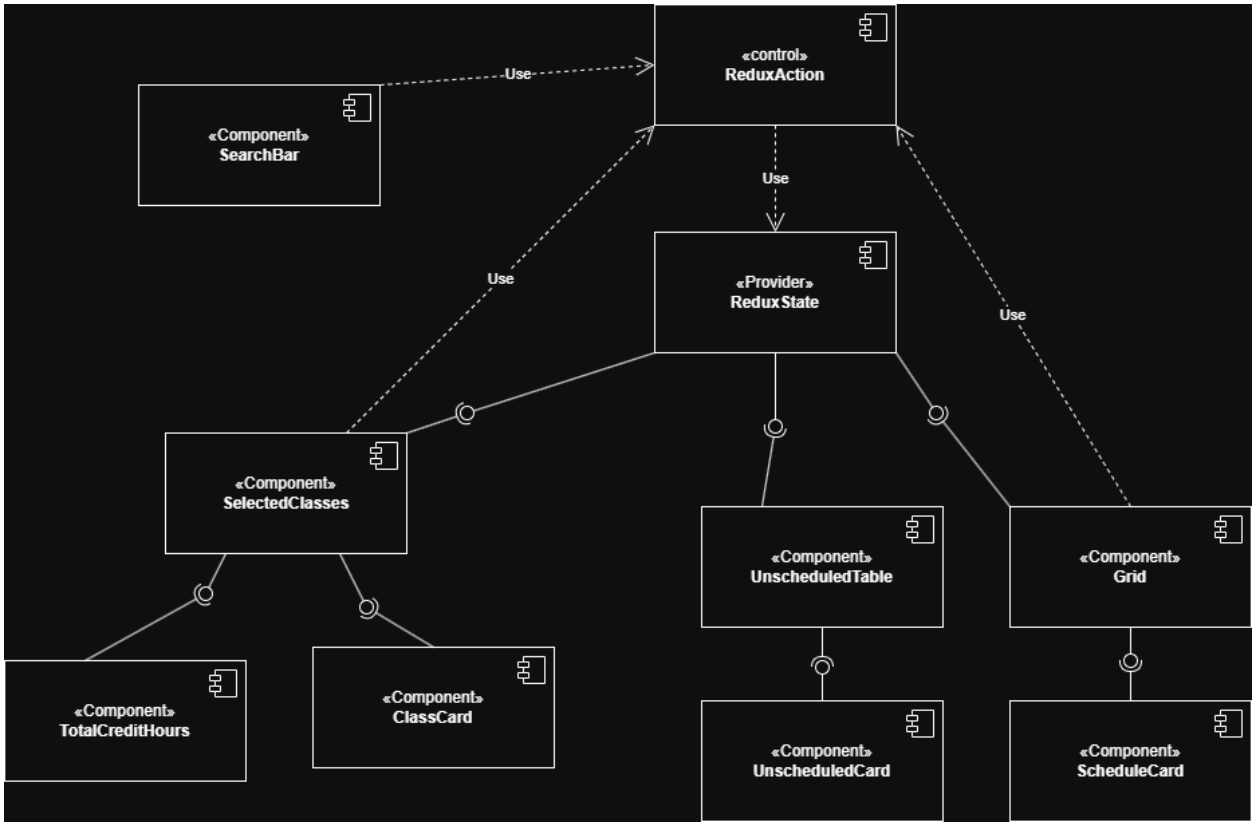
Pinning Schedules:
Within ScheduledState.permutations users can pin. When incrementing permutations the pinned card stays.

Blocking Time:
Blocked times stored in ScheduleState.blockedTimes and used in permutation logic.

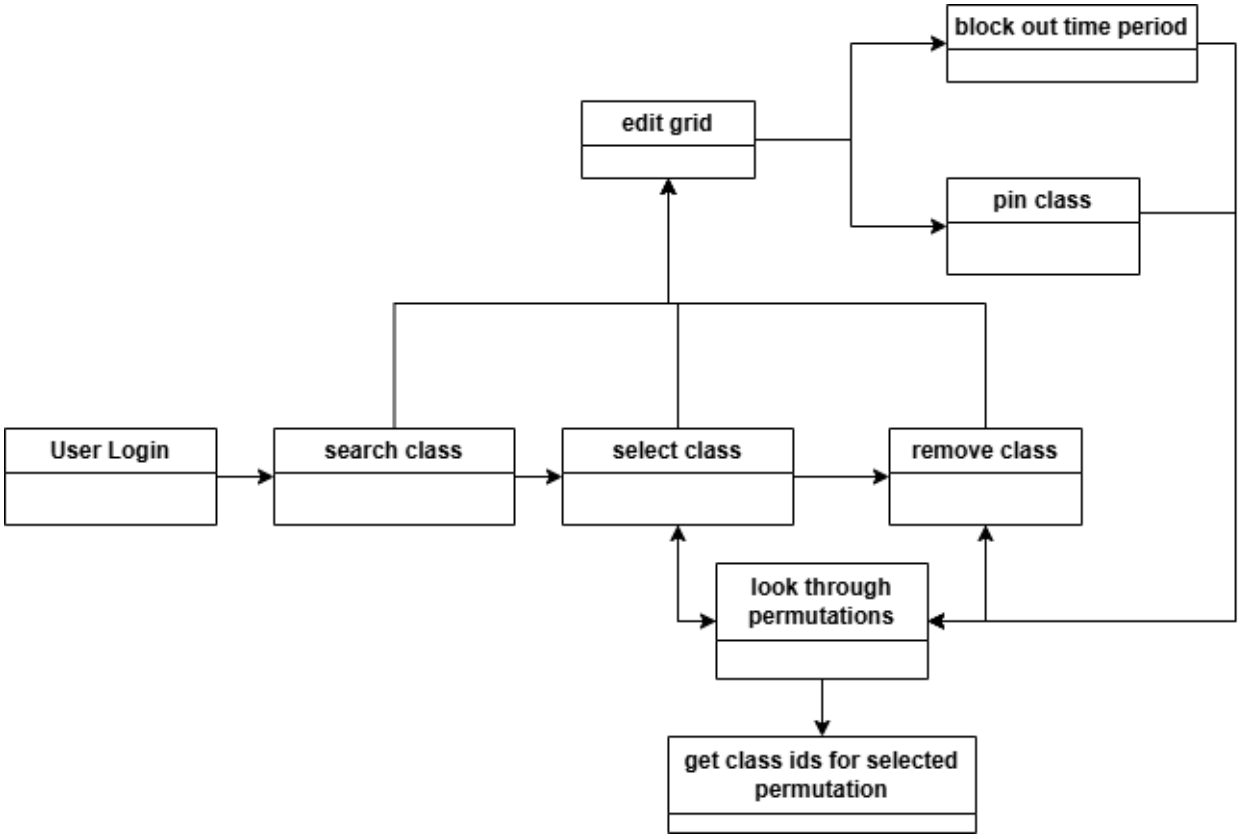
Class Diagram



Component Diagram



User Interaction Flow Diagram


block out time period
edit grid
pin class
User Login
search class
select class
remove class
look through permutations
get class ids for selected permutation