Sprint 2 Artifacts -- Update

2.4 Define Schedule Data Model

Description:
Create a database model that represents a student's schedule and its relationship to selected courses.

User Story:
As a developer, I need a Schedule model so that the backend can store and manage a user's selected courses and generated schedules.

Acceptance Criteria:

- The `Schedule` model includes a unique ID, a reference to the user (if applicable), and a list/relationship to `Course` entries. (Other fields may include created_data, last_updated_data, … etc.)
- Database migration runs successfully.
- The backend can create, read, and delete schedules.

Tasks:

- Define Schedule model in Django ORM.
- Run migrations and validate schema.
- Create test data for schedules.

Evidence of Completion:

- Screenshot of schema or model code.
- Commit: `feat: define Schedule data model and relationships`.

Testing / Verification:

- Verified in database tool (e.g., SQLite, PostgreSQL viewer).
- Tested schedule creation in backend API.

Assigned to: Connor

4.6 Select Course from List

Description:
Enable users to select a course from the course list displayed on the dashboard or semester planner page.

User Story:
As a user, I want to select a course from the list so that I can add it to my schedule.

Acceptance Criteria:

- Each listed course includes a visible “Select” checkbox (after searching is implemented, this will become an “X” to remove from schedule).
- Selecting the checkbox adds the course to the user’s selected list or state.
- Selected courses are visually distinguished.

Tasks:

- Add checkbox and event handlers for course selection.
- Maintain a selectedCourses state variable in React.
- Update UI to reflect selection.

Evidence of Completion:

- Screenshot showing selected course(s) highlighted.
- Commit: `feat: implement course selection functionality`.

Testing / Verification:

- Verified that selecting a course updates the state.
- Confirmed selected courses persist across page reloads (if backend linked).

Assigned to: Matthew

4.7 Unselect Course from List

Description:
Allow users to deselect a previously selected course.

User Story:
As a user, I want to unselect a course from my list so that I can modify my planned schedule easily.

Acceptance Criteria:

- "Unselect" action removes the course from the selected list.
- UI immediately reflects the unselection.
- State updates correctly in memory or backend.

Tasks:

- Add a handler for unselect action.
- Update selectedCourses array to remove item by ID.
- Adjust UI styling for deselection.

Evidence of Completion:

- Screenshot of UI showing course removed from selected list.
- Commit: `feat: allow unselecting of selected courses`.

Testing / Verification:

- Verified by toggling multiple courses on and off.
- Confirmed no state or rendering errors occur.

Assigned to: Matthew

4.8 Form All Possible Schedules

Description:
Generate all valid combinations of selected courses that do not conflict by time.

User Story:
As a user, I want the system to generate all possible schedule combinations so that I can choose the one that fits best.

Acceptance Criteria:

- Given a list of selected courses, the backend computes all valid non-conflicting combinations.
- Each schedule includes a list of courses and time blocks.
- Handles cases where conflicts make no valid schedules.

Tasks:

- Implement schedule generation algorithm (e.g., backtracking or combinatorial filtering).
- Ensure time conflicts are correctly detected.
- Response to the frontend is correctly formatted.

Evidence of Completion:

- Screenshot of console output or API response showing multiple schedule combinations.
- Commit: `feat: generate all valid schedules from selected courses`.

Testing / Verification:

- Verified with test datasets (e.g., overlapping vs. non-overlapping courses).
- Confirmed correct number of schedules generated.

Assigned to: Connor, Mattew

5.1 Create Schedule Page Layout

Description:
Design the layout where users can view, modify, and visualize their selected schedules.

User Story:
As a user, I want a Schedule page that clearly displays my generated schedules and selected courses.

Acceptance Criteria:

- Page layout follows consistent navigation structure.
- Includes sections for course list, selected courses, and schedule visualization.
- Responsive and readable design.

Tasks:

- Display courses as a list of cards -- Sidebar Component (5.6).
- Create `SchedulePage.jsx` component.
- Use grid/flex layout for main sections.
- Integrate placeholders for sidebar and calendar.

Evidence of Completion:

- Screenshot of Schedule page layout.
- Commit: `feat: create schedule page layout`.

Testing / Verification:

- Manually verified visual layout in browser.
- Checked responsiveness.

Assigned to: Bryson

5.2 Create Dashboard Page Layout

Description:
Design a dashboard where users can access app features and navigate to schedule builder, profile, etc.

User Story:
As a user, I want a dashboard page that gives me an overview of available actions and quick access to key features.

Acceptance Criteria:

- Dashboard displays welcome text, navigation links, and recent activity or saved schedules.
- Uses consistent styling with other pages.

Tasks:

- Create `DashboardPage.jsx`.
- Add navbar or sidebar with navigation links.
- Add placeholders for content sections.

Evidence of Completion:

- Screenshot of Dashboard layout.
- Commit: `feat: implement dashboard page layout`.

Testing / Verification:

- Verified page loads and links work correctly.

Assigned to: Owen, Bryson

5.3 Create Login Page Layout

Description:
Implement a login interface with input fields and authentication triggers.

User Story:
As a returning user, I want to log into my account so that I can access my saved schedules.

Acceptance Criteria:

- Login form includes email and password fields.
- Includes the “Login” button.
- Includes the "Sign Up” button.
- Includes the "Continue as Guest” button.

Tasks:

- Create `Login.tsx` with form fields.
- Connect to backend login endpoint (if implemented).

Evidence of Completion:

- Screenshot of login form.
- Commit: `feat: add login page layout and form`.

Testing / Verification:

- Verified form input behavior manually.
- Confirmed navigation to dashboard after login (if backend auth available).

Outcome:
✅ Login page layout designed and functional.

5.4 Create Sign Up Page Layout

Description:
Provide a registration page for new users.

User Story:
As a new user, I want to create an account so that I can start building my schedules.

Acceptance Criteria:

- Signup form includes name, email, password, and confirmation password.

Tasks:

- Create `Signup.tsx`.
- Add validation logic.
- Link to backend registration endpoint.

Evidence of Completion:

- Screenshot of sign-up form.
- Commit: `feat: implement sign-up page layout`.

Testing / Verification:

- Manually tested valid and invalid inputs.
- Verified success navigation.

Outcome:
✅ Sign-up page layout complete and functional.

5.6 Selected Classes Shown in a Sidebar (part of 5.1)

Description:
Display the user's selected courses in a sidebar component for quick access and review.

User Story:
As a user, I want to see my selected courses in a sidebar so that I can easily track which classes are currently in my schedule.

Acceptance Criteria:

- Sidebar updates dynamically as courses are selected/unselected.
- Displays course name, code, and time.
- Collapsible or scrollable design.

Tasks:

- Create `Sidebar.jsx` component.
- Bind to selectedCourses state.
- Style for clarity and compactness.

Evidence of Completion:

- Screenshot showing sidebar with selected courses.
- Commit: `feat: add sidebar displaying selected courses`.

Testing / Verification:

- Verified dynamic updates upon selection/unselection.
- Checked layout responsiveness.

Outcome: Bryson, Connor

5.7 Weekly Calendar Shows Selected Courses at Their Times

Description:
Render selected courses visually on a weekly grid calendar by their day/time slots.

User Story:
As a user, I want to see my classes displayed on a weekly calendar so that I can visualize my schedule.

Acceptance Criteria:

- Calendar grid shows Monday–Friday columns.
- Courses displayed as time blocks at correct positions.
- Overlapping courses detected and visually distinguished.

Tasks:

- Implement or import calendar component.
- Map course time data to grid coordinates.
- Style events for readability.

Evidence of Completion:

- Screenshot of weekly calendar with displayed courses.
- Commit: `feat: display selected courses on weekly calendar`.

Testing / Verification:

- Verified correct time positioning.
- Tested overlapping courses visually.

Assigned to: Mohamed

5.9 View All Possible Schedules

Description:
Allow users to view all generated non-conflicting schedules in a navigable interface.

User Story:
As a user, I want to browse all possible valid schedules so that I can select the one I prefer most.

Acceptance Criteria:

- Displays all generated schedules, one at a time or in a scrollable list.
- User can cycle or click through them.
- Selected schedule is clearly highlighted.

Tasks:

- Create component to render multiple schedules.
- Add buttons or navigation to switch between schedules.
- Integrate with backend schedule generation logic.

Evidence of Completion:

- Screenshot of UI showing multiple schedule views.
- Commit: `feat: implement viewing of all possible schedules`.

Testing / Verification:

- Verified with 3+ test combinations.
- Confirmed accurate rendering and navigation.

Assigned to: Connor, Owen

Source: Initial generations by ChatGPT | Edited by Group 4