Speaker Notes for Demo 2

Slide 1: Introduction

- StressLess is a semester-long project for CSC-324, created by Khanh Do, Nifemi Ogunmesa, Cheyanne Vinson, Madel Sibal, Alyssa Trapp, and Tim Yu.

Slide 2: Product Overview

- Description of StressLess:
    - StressLess is a smart schedule tailored to an individual's productivity preferences. The user can input their schedule and answer a preference survey, and StressLess will customize the best possible schedule for them based on their input.

Slide 3: Software Architecture

- Implemented:
    - Frontend:
        - Authentication service
        - Sign up
        - Sign in
        - Sign out
    - Backend:
        - CRUD operations for auth
        - CRUD operations for events
        - CRUD operations for deadlines
        - CRUD operations for preferences
    - Database:
        - Writing and reading database (via Prisma)
        - Database schema implemented

Slide 4: Database Schema

- All the tables are created already in accordance to the schema.
- We use Prisma to create tables from the schema.
- The database uses PostgreSQL hosted by Neon.

Slide 5: Demo and Code Explanation

- We will first show the new updates in our frontend, which is integrated with real API call to our backend. We also wrote unit tests for our backend API endpoints. We will also showed our Continuous Deployment pipeline.

Slide 6: New Features

- We will explain and demonstrate the new features we implemented.

Slide 7: User Preferences

- User's preferences answers are shown on Profile Page. Before, it uses default values. Now, it fetches the data from the backend to show on the frontend.
- When an user edit their preferences and save their new preferences, the database is updated with the new answers. When the users come back to their Profile Page, that new information replaced the old one.

Slide 8: Adding Events

- User's preferences answers are shown on Profile Page. Before, it uses default values. Now, it fetches the data from the backend to show on the frontend.
- When an user edit their preferences and save their new preferences, the database is updated with the new answers. When the users come back to their Profile Page, that new information replaced the old one.

Slide 9: Sidebar

- There is now a side-bar (or called hamburger menu) that centralizes all routes that we have: Calendar, Profile Page. Upon signed-in, user is routed to /dashboard where they can click on the side-bar button to show navigations to other places of the app.

Slide 10: Logout & Protected Routes

- There is now a Log out button that log the user out.
- If the user is not signed-in (or not authenticated), they can only route to / . Meaning if they try to go to /dashboard, they will be automatically redirected.
- If the user is signed-in (or is authenticated), they will not see the Sign up, Sign in form on the landing page /, but they will have the option to click on a button that says Go to dashboard.

Slide 11: Help Button

- There is now a Help button on the top right corner of the app. Upon clicking, a pop-up will appear that says how to use the app in detailed.

Slide 12: Continuous Deployment

- All commits to the main branch automatically trigger re-deployment via GitHub Actions.
- Both frontend and backend are hosted on Render, ensuring live updates with every merge.

Slide 13: Backend: Events & Deadlines

- API Endpoints Create-Read-Update-Delete (CRUD) operation for events and deadlines, ready to be integrated end-to-end.

Slide 14: Demonstration - Unit Tests

- Note: Screenshots on the slide are for reference only. Code will be demonstrated live during the presentation.
- The left side shows a test for one of the user routes, which illustrates a user preference being retrieved from the database
    - To test the route, we first declare an array of user preferences, which we then populate with fake data and lastly push to the database
    - Then, we retrieve the user preference by its ID to check if the get route worked
    - Lastly (not shown in the screenshot), we delete our data from the database following a successful test
- The right side shows the starter code for our calendar tests, which shows all the declarations for our calendar tests, later populated with fake IDs following their declarations

Slide 15: Demonstration - Fake Data Factory & Our Database

- Note: Screenshots on the slide are for reference only. Code and database will be demonstrated live during the presentation.
- To test if the data is being stored in our database, we set up and utilized a fake data factory to populate the columns of each of our data models.
    - The left side shows the code for the set up of the fake data factory.
- Due to usage constraints on Neon database, we had to create a separate testing branch in our database for automated testing purposes only. This is so that we could ensure that we do not exceed the usage limits of our database and render the database unusable.
    - The right side shows the Neon database interface, and the top left corner shows the branches in our database.

Slide 16: Q&A

- Ask the class if they have any questions