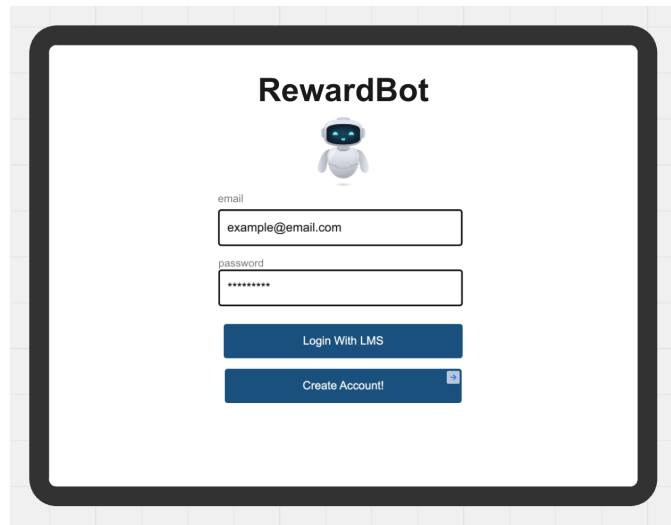


1. Process Deliverable: RewardBot Prototype

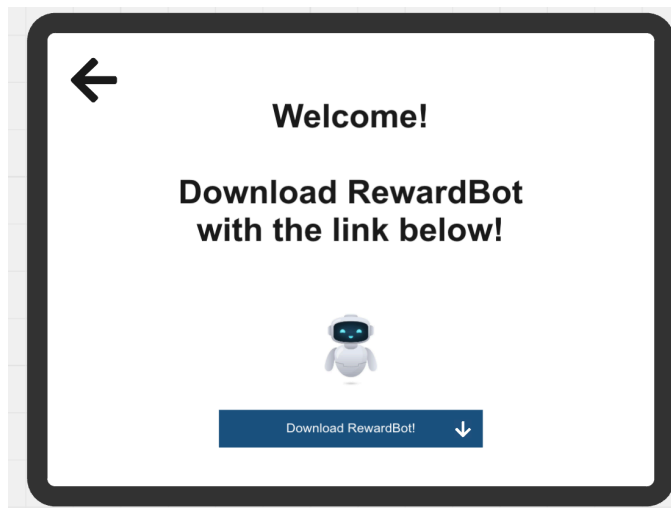
This is the prototype for the RewardBot system. It was implemented using Miro. It goes through the system and its integration with Canvas and how the system changes based on completion of assignments.

RewardBot Login



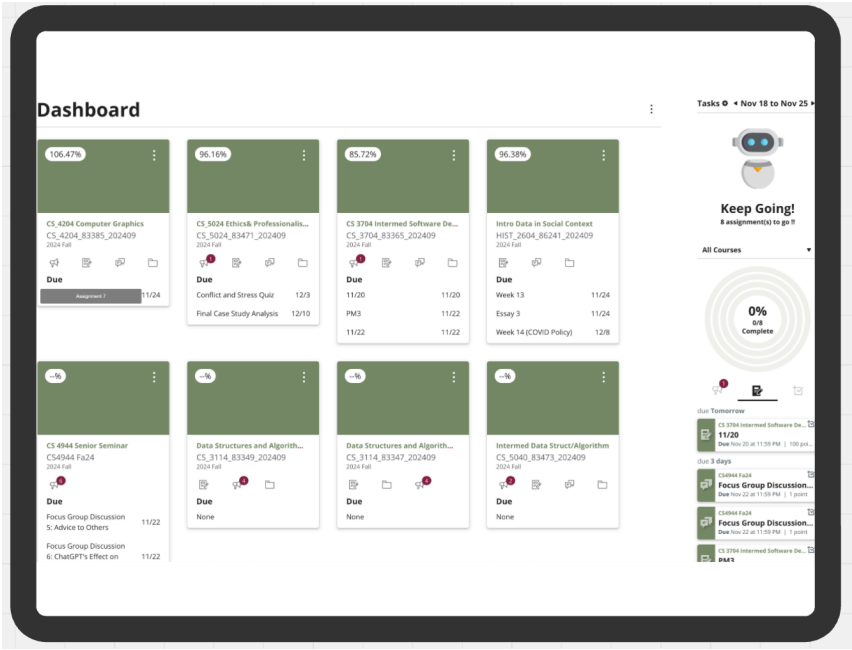
The login screen for RewardBot features a white background with a black border. At the top center is the text "RewardBot" in bold, followed by a small robot icon. Below the icon are two input fields: the first is labeled "email" and contains the text "example@email.com"; the second is labeled "password" and contains a series of asterisks. At the bottom are two blue buttons: "Login With LMS" and "Create Account!".

RewardBot Download

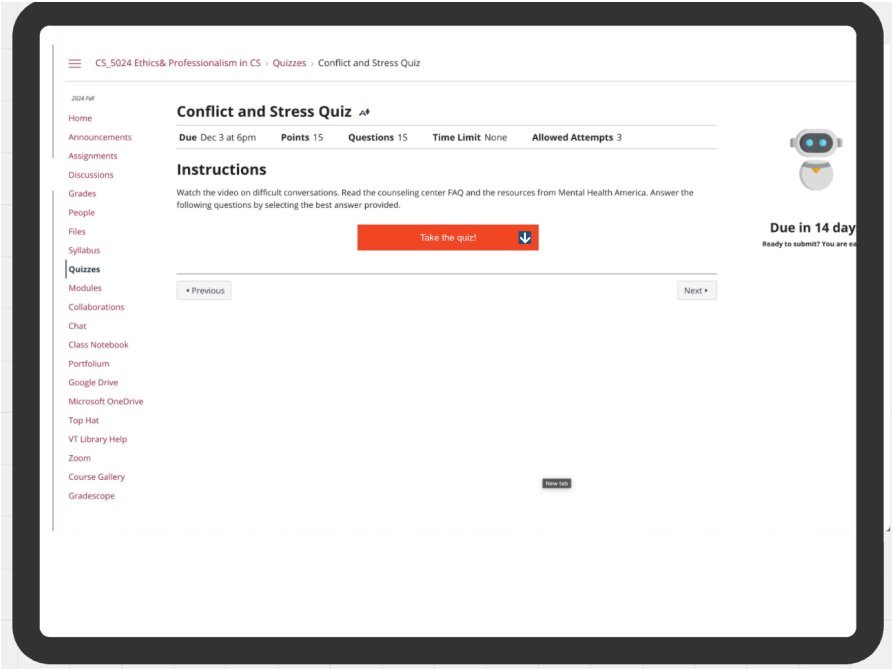


The download screen for RewardBot features a white background with a black border. In the top left corner is a black back arrow. In the center is the text "Welcome!" in bold, followed by "Download RewardBot with the link below!" in bold. Below the text is a small robot icon. At the bottom is a blue button with the text "Download RewardBot!" and a downward arrow icon.

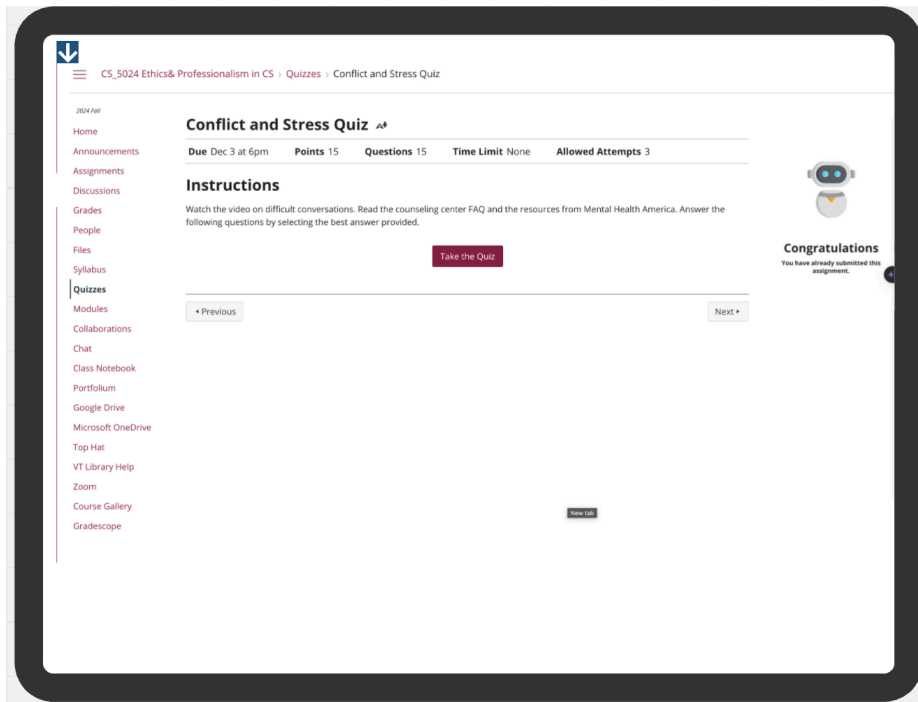
Canvas RewardBot Integration, assignments not completed



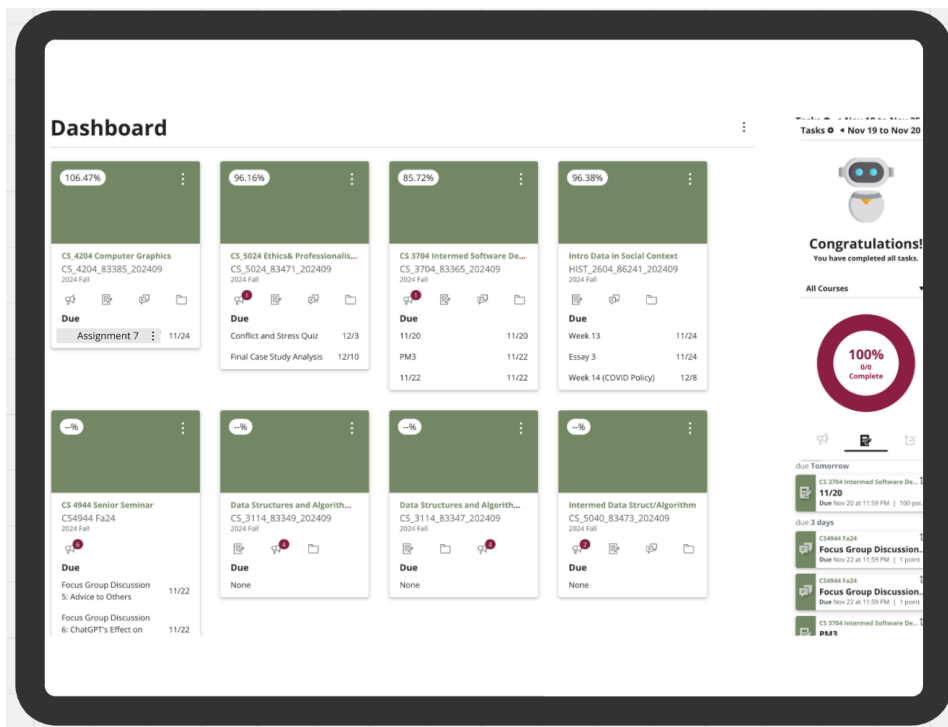
Assignment not finished in Canvas



Canvas assignment completed



Canvas RewardBot after all assignments completed



2. High Level Design

For our project, we will be using the layered architectural pattern, specifically the MVC (Model-View-Controller) model. To implement this, we have chosen the MERN (MongoDB, Express.js, React.js, Node.js) technology stack. According to the official MongoDB website, MERN is a 3-tiered architecture consisting of a front-end built with React.js to create the user interface, a back-end using Express.js and Node.js to handle HTTP requests and communicate with the front-end, and a database tier, where MongoDB will be used to store and manage data.

We decided to use the layered 3-tier architecture because of team contribution and modularity. First, since the purpose of our project is to build an application that stores, processes, and visually displays data to users, a web application was the most suitable choice. Additionally, one of our team members has experience building web applications with the MERN stack, and React is relatively straightforward to learn and implement, allowing all team members to effectively contribute. Second, the MVC architecture offers significant flexibility and maintainability. By separating the Model, View, and Controller layers, we can independently modify or replace components without overhauling the entire system. For example, while we initially chose MongoDB for the database, it is possible to switch to MySQL or another database system with minimal changes to the rest of the application. This flexibility also extends to supporting multiple views for a single dataset, which can be useful if we expand to mobile or other platforms in the future.

3. Low Level Design

Since we are using React for our front-end we will mainly focus on the Behavior Design Pattern Family, specifically the State Pattern. React component's are objects that will be created and reused throughout the program. Due to the React's declarative and state-driven architecture it will mostly use this pattern.

In each step in the component we would use the useState hook to represent different states of the component, and the component will dynamically switch between these steps and render a different UI element accordingly.

Example code:

The useAuth custom hook will load the user from the database and save it in the state. However if the information of the user changes, the UI will render again and reload it with new information.

useAuth.jsx:

```
import { apiGetUser } from "../../api/user";
import { useEffect, useState } from "react";
import { validateToken, getCanvasUser } from "../../api/canvas";
import { useNavigate } from "react-router-dom";

export const useAuth = () => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);
```

```

useEffect(() => {
  const fetchUser = async () => {
    try {
      const data = await apiGetUser();
      setUser(data.user);
    } catch (error) {
      console.error(error);
    } finally {
      setLoading(false);
    }
  };

  fetchUser();
}, []);

return {
  user: {
    username: user?.username,
    email: user?.email,
    password: user?.password,
    canvasToken: user?.canvasToken,
    createdAt: user?.createdAt,
  },
  loading,
};
};

export const useCanvasAuth = () => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);
  const navigate = useNavigate();

  useEffect(() => {
    validateToken(
      getCanvasUser,
      (data) => {
        setUser(data);
        setLoading(false);
      },
      () => {
        navigate("/login");
      },
    );
  });
};

```

```

    );
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  return {
    canvasUser: {
      image: user?.avatar_url,
      firstname: user?.first_name,
      id: user?.id,
      lastname: user?.last_name,
    },
    loadingCanvas: loading,
  };
};

```

Profile.jsx

```

import React from "react";
import { useAuth, useCanvasAuth } from "../../hooks/auth/useAuth";

const ProfilePage = () => {
  const {
    user: { email, createdAt },
    loading,
  } = useAuth();
  const {
    canvasUser: { image, firstname, lastname },
    loadingCanvas,
  } = useCanvasAuth();

  if (loading || loadingCanvas) {
    return <div>Loading...</div>;
  }

  return (
    <div className="flex min-h-screen items-center justify-center bg-gray-100">
      <div className="w-full max-w-sm rounded-lg bg-white p-6 shadow-md">
        <div className="flex flex-col items-center">
          <img
            className="h-24 w-24 rounded-full border-4 border-blue-500"
            src={image ?? "https://via.placeholder.com/150"}

```

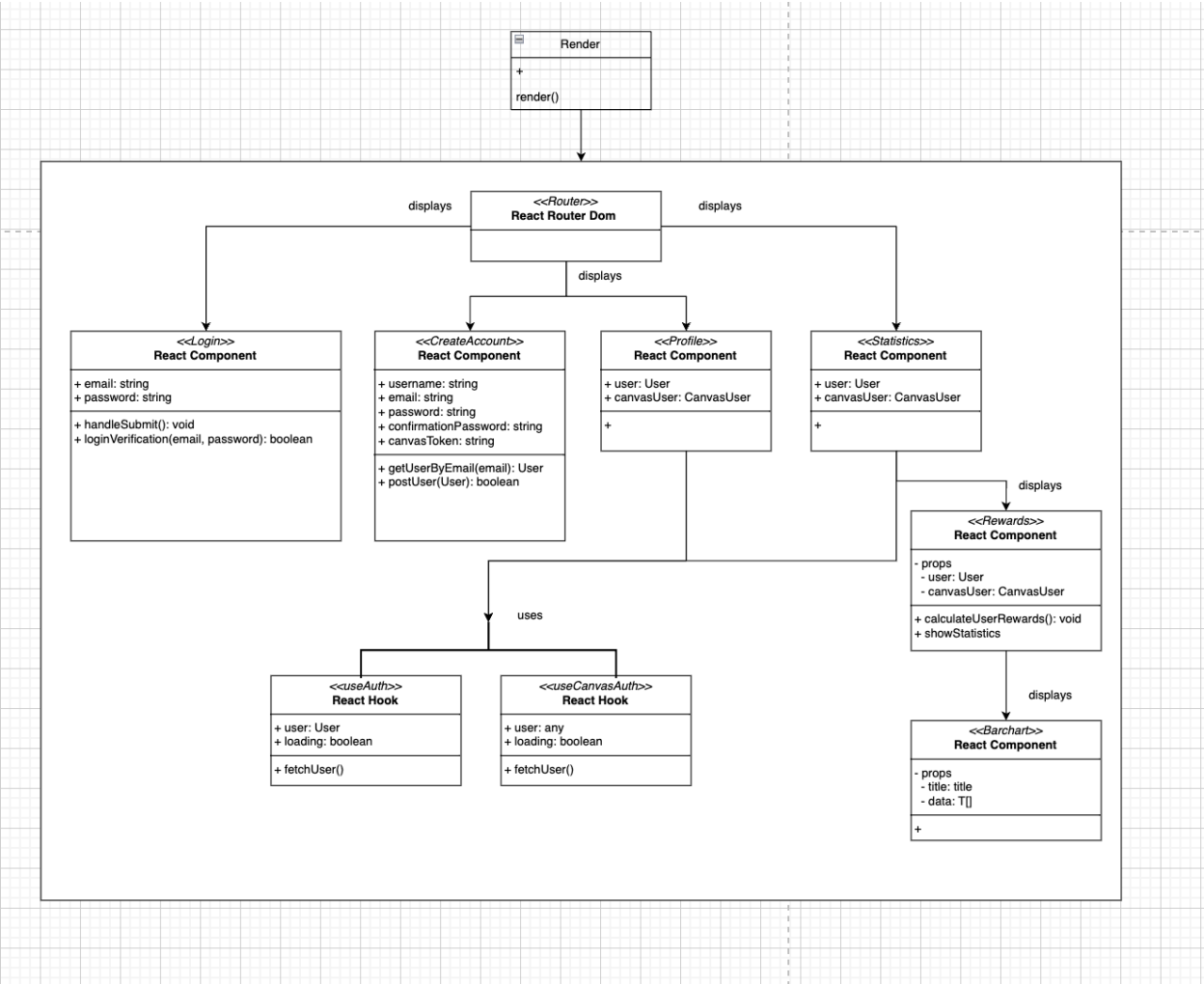
```

        alt="User Avatar"
      />
      <h2 className="mt-4 text-2xl font-bold text-gray-800">
        {firstname} {lastname}
      </h2>
    </div>
    <div className="mt-6 space-y-4">
      <div className="flex items-center justify-between">
        <span className="text-gray-600">Email:</span>
        <span className="font-medium text-gray-800">
          {email}
        </span>
      </div>
      <div className="flex items-center justify-between">
        <span className="text-gray-600">Joined:</span>
        <span className="font-medium text-gray-800">
          {createdAt}
        </span>
      </div>
    </div>
  </div>
</div>
);
};

export default ProfilePage;

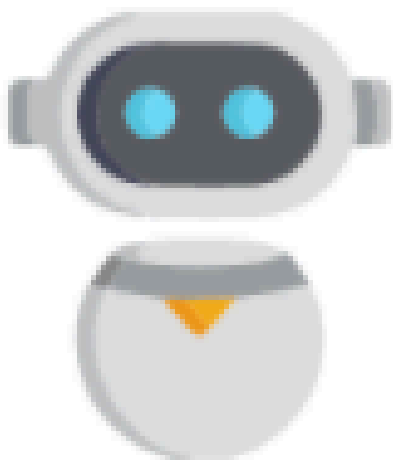
```

Informal class diagram:



4. Design Sketch

This is the sketch of our robot. It will be integrated into Canvas. As seen above, we have coded a login page, and integrated the robot into the canvas home page and the canvas assignment submission page.



←

→

November 2024

Week

Month

Agenda

+

SUN	MON	TUE	WED	THU	FRI	SAT
27	28	29	30	31	1	2
	<div>11:25a Attendance...</div> <div>3:20p a1028</div> <div>10/28 ICS_3704_8...</div> <div>Project 2B - Dream...</div>	<div>10/29</div> <div>10/30</div>	<div>3:20p a1030</div> <div>11p Project 3 Milest...</div> <div>10/30 ICS_3704_8...</div>	<div>10/31</div> <div>Quiz5</div>	<div>3:20p a1101</div> <div>11/1 ICS_3704_83...</div> <div>PM42</div>	
3	4	5	6	7	8	9
	<div>11:25a Attendance...</div> <div>3:20p a1104</div> <div>11p Project 3 Milest...</div> <div>11/4 ICS_3704_83...</div>	<div>10/34</div>	<div>3:20p a1106</div> <div>11/6 ICS_3704_83...</div>	<div>10/35</div> <div>Quiz6</div>	<div>3:20p a1108</div> <div>11p Project 3</div> <div>11/8 ICS_3704_83...</div>	
10	11	12	13	14	15	16
	<div>9a Exam2A</div> <div>11:25a Attendance ...</div> <div>3:20p a1111</div> <div>11/11 ICS_3704_8...</div> <div>Project 2C - Dream...</div> <div>Quiz 3: Databases...</div>		<div>3:20p Exam2B</div> <div>11/13 ICS_3704_8...</div>		<div>3:20p a1115</div> <div>11/15 ICS_3704_8...</div> <div>Homework-4</div> <div>Project 4 Milestone 1</div>	
17	18	19	20	21	22	23
	<div>11:25a Attendance ...</div> <div>2:30p 11/18 ICS_37...</div> <div>3:20p a1118</div> <div>Discussion 2: Andro...</div>	<div>10/40</div>	<div>3:20p a1120</div> <div>11/20 ICS_3704_8...</div> <div>Quiz7</div>	<div>10/44</div>	<div>3:20p a1122</div> <div>11/22 ICS_3704_8...</div> <div>PM3</div> <div>Project 4 Milestone 2</div>	
24	25	26	27	28	29	30
					<div>VT SPOT</div>	

< November 2024

27	28	29	30	31	1
3	4	5	6	7	8
10	11	12	13	14	15
17	18	19	20	21	22
24	25	26	27	28	29

CALENDARS

Lakshitha Gattu

Cryptography_87244

CS 3714 Fall 24 Android Mobile Development

Data Structures and Algorithms Hamouda

Honors Credit Tracker 2023/2024 Cohort

Intermed Software Des

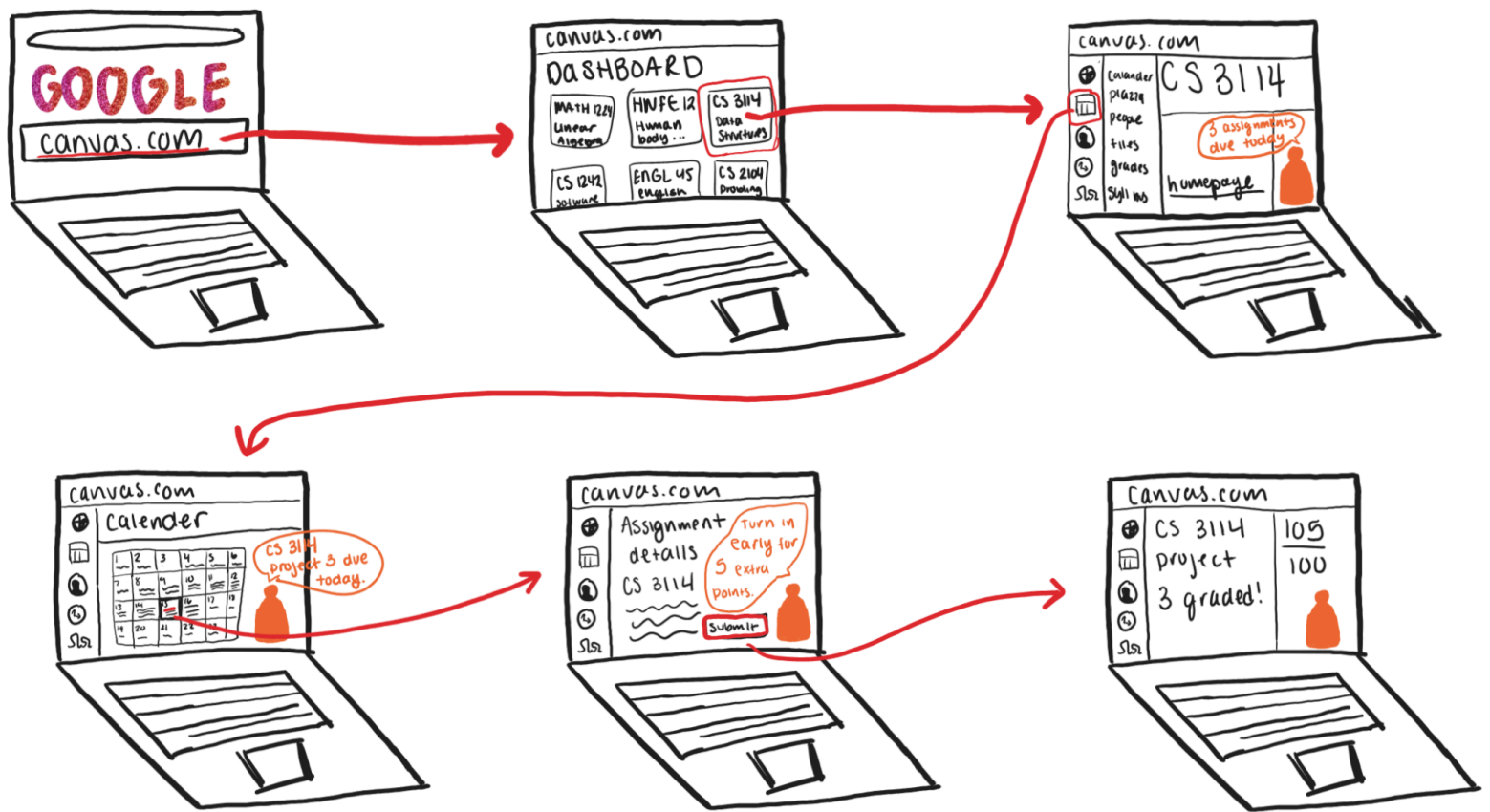
OpenDSA Textbook for CS 3114/5040 Fall 2024

Softw Des & Data

Placeholder

4 assignments due today

Above is the wireframing for the calendar page. The wireframe was designed with usability principles such as affordance and visibility in mind. The calendar grid is the primary focal point, ensuring users can intuitively locate dates and events at a glance. By placing the robot prominently in the sidebar, it is both noticeable and accessible, adhering to the principle of information prioritization. The clean, minimalistic design aligns with the aesthetic usability effect, making the interface feel approachable while still being functional.



Above, you can see that the robot is positioned in the bottom-right corner across several key pages: the course homepage, the calendar page, the assignment details page, and the assignment submission page. This consistent placement ensures that the robot is always accessible, providing users with real-time assistance and updates no matter where they are in the interface. Its visibility across these pages reinforces its role as a helpful companion, streamlining navigation and enhancing the overall user experience.