Group 1: Lisa Heinzman, Shakeera Singh, Sara Schwartz, Ja'Mecia Rosier, Jacob Rothstein

Discovery Phase

DIG 4104C - 0077/0177

Prof Novatnak

Statement of Work

What is the project about?

Our project concept is to create a simple mobile application that contains helpful guides for everyday tasks and give users a way to organize their schedules and to - do lists. Users will be able to have their own profile, which allows them to create events and tasks in the calendar and tasks part of the application that are linked to their personal account. We want to create an accessible app that will provide users with four different color palettes and will ensure that our text is readable.

"Hive Help" is not a redesign, but is a new application that pulls features from various scheduling, lifestyle and time management applications. We have taken into account apps with similar features and have tried to make a new application where everything is at your fingertips. Our application will feature various pages for the user to navigate: Guides (which is broken down into three categories: School, Work, and Personal), Tasks, Calendar, Profile, and a central Home page.

Who is on your team, and what are their duties?

- **Ja'Mecia Rosier - Front-End Developer:**
    - Duties:
        - Front-end development of the mobile application.

        - Implementing the user interface design, including navigation, widgets, and overall visual elements.
        - Collaborating with the front-end designer to ensure a cohesive and user-friendly interface.
        - Participating in user testing to gather feedback on the usability of the front-end.
- **Lisa Heinzman - Back-End Developer:**
    - Duties:
        - Tasked with the back-end development.
        - Creating and managing the database using MySQL.
        - Utilizing React and Node to develop the application
        - Collaborating with front-end developers to establish seamless communication between the front and back ends.
        - Implementing a secure and efficient back-end framework using Express and NodeJS.
- **Sara Schwartz - Front-End Designer:**
    - Duties:
        - Tasked with the front-end design.
        - Creating a visually appealing and user-friendly interface based on the project's design and style guide.
        - Collaborating with the front-end developers to ensure design concepts are implemented successfully.
        - Conducting user testing with a focus on the application's graphic elements.

- Sh**akeera Singh - Front-End / Back-End Dev:**
    - Duties:
        - Involved in both front-end and back-end development.
        - Assigned the role of guide writer and additional responsibility in database development.
        - Utilizing React and Node to develop the application
        - Collaborating with front-end developers to establish seamless communication between the front and back ends.
        - Writing guides for the application and ensuring integration with the database.
- **Jacob Rothstein - UI/UX Researcher:**
    - Duties:
        - Focused on user interface and user experience research.
        - Conducting research on competitor applications and presenting insights to the team.
        - Contributing to the development of user interface prototypes.
        - Collaborating to create both a hi-fi and lo-fi prototype in Adobe XD.
        - Participating in user testing to refine and improve the user experience.

Is the project an innovation, a redesign of something existing, or a redevelopment of an idea?

Our project is a redesign of existing applications on the market. However, it is a redevelopment of multiple ideas combined into one. So while we are not directly remaking a

single app, we are taking features we have used personally and combining them so that it is easier for users to have their organizational needs in one place.

Is it suited better as a mobile app or website?

The project is suited better for a mobile app due to the need for quick accessibility. The target users being students and young adults would likely benefit from the application being readily available on their phones or tablets.

Funding Plan

Our plan for funding our app is to have the app available for free, with occasional 15-second ads. For the user to remove the ads, they will pay a $2.99 monthly subscription in their device's app store. If the user cancels this subscription plan, they will be moved back to the free version with ads when their monthly plan ends.

To additionally help fund our app, we will seek out companies to partner with in sponsored guide pages. The companies will provide guides (available for a limited-time, based of the length of the contract) for users based on their area(s) of expertise. Examples of potential companies to collaborate with include: HelloFresh and BetterHelp. HelloFresh's collaboration could include a guide on how to prepare and cook basic meals/meals from them, and BetterHelp could include a guide on how to access therapy through them.

Estimated salary per team member:

- Ja 'Mecia Rosier: $46,080
- Lisa Heinzman: $73,440
- Sara Schwartz: $40,320

- Shakeera Singh: $51,840
- Jacob Rothstein: $51,840

Projected production costs:

INCOME		Budget
Internal Funding After 1 year		
	Premium Tier Purchases (Projected 1000 purchases a month at $2.99)	233,220
	Ad Revenue (Projected $200 per day)	73,000
	Sponsored Content (Projected $1000 per sponsor 4x/month)	30,000
	Other	
	Total Internal Income	**336,220**
External Funding/Other		
	Government Grants	10,000
	UCF Blackstone Launchpad Grant	5,000
	Kickstarter Campaign	20,000
	Total External Income	**35,000**
Total INCOME		**371,220**

EXPENSES		Budget
Software and Licensing		
	Adobe XD subscription (9 months)	90
	Apple Developer account (1 year)	99
	Android App Store fee	25
	Microsoft Store fee	99
	Subtotal	**313**
Salaries	9 months at 40 hours per week	
	Social Media Intern	25,000
	Ja 'Mecia Rosier	46,080
	Lisa Heinzman	73,440
	Sara Schwartz	40,320
	Shakeera Singh	51,840
	Jacob Rothstein	51,840
	Subtotal	**288,520**
Total EXPENSES		**288,833**

NET (Income - Expenses)	**82,387**

- Link to Budget Spreadsheet: [Budget Sheet]

Strategy Guide

The application that our group will be creating for this semester is a scheduling and lifestyle application. As students, all of us struggle with the daily juggling act of managing our part-time jobs, social relationships and our educational duties. This is what inspired us to choose this breed of application for our project. The application, named “Hive Help”, will be targeted toward users ages 18-35, for our society’s students and young adults. We intend to assist our audience to better organize their day-to-day schedules and tasks, as well as provide guides for self-care, work and school.

Our application will help students organize their day-to-day schedules and tasks, and provide life tips, self-care tutorials, and other guides to help them succeed in school and life challenges. Our project’s goals are to make an app that is beneficial for our target audience, solve problems frequently seen in competitor’s apps, and create a product that stands out from others in the market. The competitor apps that we covered in our research were

During the research for our case studies, we made large strides in deciding what features we felt worked for this genre of application. Each of us five collected a list of different competitor applications that we felt related to the kind of application that we wanted to create. The competitors that we researched were AloeBud, RealWorld, Notion, Evernote and Time Tree. Drawing inspiration and insights from these applications, we aim to amalgamate their strengths while addressing their respective shortcomings.

AloeBud's unique pixel-art icons and gentle nudges for self-care tasks stand out as commendable features. However, the absence of crucial accessibility features, such as light/dark mode and text size adjustments, poses a significant challenge. Limited customization options for contrast settings further underscore the need for improvements.

RealWorld distinguishes itself by offering guides and step-by-step instructions for navigating the challenges of adulthood. However, its game-like structure and reliance on paywalls hinder user interaction. Additionally, the lack of interactive features in the "activity" section presents room for enhancement.

TimeTree's intuitive calendar application serves as a benchmark for scheduling simplicity. Catering to both individual and group users, it adeptly manages varying schedules and time zones. Nevertheless, its primary focus on calendar features and limited interaction in the "memo" and "activity" sections necessitate a more comprehensive approach.

Notion's versatility in creating checklists, notes, wikis, and projects is commendable. However, its plain and unappealing interface, coupled with limited differentiation from default iPhone apps like "Reminder" and "Notes," calls for a more visually engaging design.

Evernote’s capabilities in providing a powerful tool in assisting users with note-taking and to-do lists is an inspiration to our application. However, we intend to take those assets and embellish them by creating more customization options to make it more engaging for the user.

Our new application seeks to strike a balance by synthesizing the strengths of AloeBud, RealWorld, TimeTree, Evernote and Notion while addressing their deficiencies.

- Calendar Feature: Inspired by TimeTree's intuitive calendar, our application will introduce a user-friendly calendar for both individual and group use.

- To-Do List: Recognizing the limitations in TimeTree, our application will feature a dedicated to-do list for efficient task management.

- Help Guides: Learning from RealWorld's emphasis on guides, our application will offer comprehensive guides for self-care, work-life balance, and schoolwork management.

- Note-Taking: The ability to create efficient notes throughout our application like used by Evernote will provide a more personal and engaging way for users to manage their events and tasks.

- Customization: Addressing AloeBud's lack of customization options, our application will prioritize personalized layouts, color schemes, and features.

- User-Friendly Design: Acknowledging Notion's plain interface, our application will focus on an aesthetically pleasing and easy-to-use design.

- Accessibility: Drawing from AloeBud's shortcomings, robust accessibility features will be integrated, encompassing light/dark mode options, text size adjustments, and customizable contrast settings.

Our new application aims to transcend the boundaries set by existing tools, creating a user-centric, inclusive, and versatile platform. By assimilating the strengths and addressing the weaknesses of AloeBud, RealWorld, TimeTree, Evernote and Notion, we endeavor to craft a holistic solution that enhances organization, self-care, and overall lifestyle management for our target audience. Through this amalgamation, we aspire to redefine the landscape of lifestyle applications, providing a seamless and enriching experience for users navigating the complexities of modern life.

An integral feature of our concept is to include assisting guides that help our target audience with important information and tips that will aid them in their personal and professional endeavors. We plan to source this information with the help of AI tools such as ChatGPT to aid in the initial collection of this educational information. We will then parse through the results manually and make the necessary corrections using research.

The testing strategy encompasses functionality, performance, and usability testing, addressing potential risks such as selecting the right user sample and ensuring users possess minimal pre-existing knowledge of the application interface.

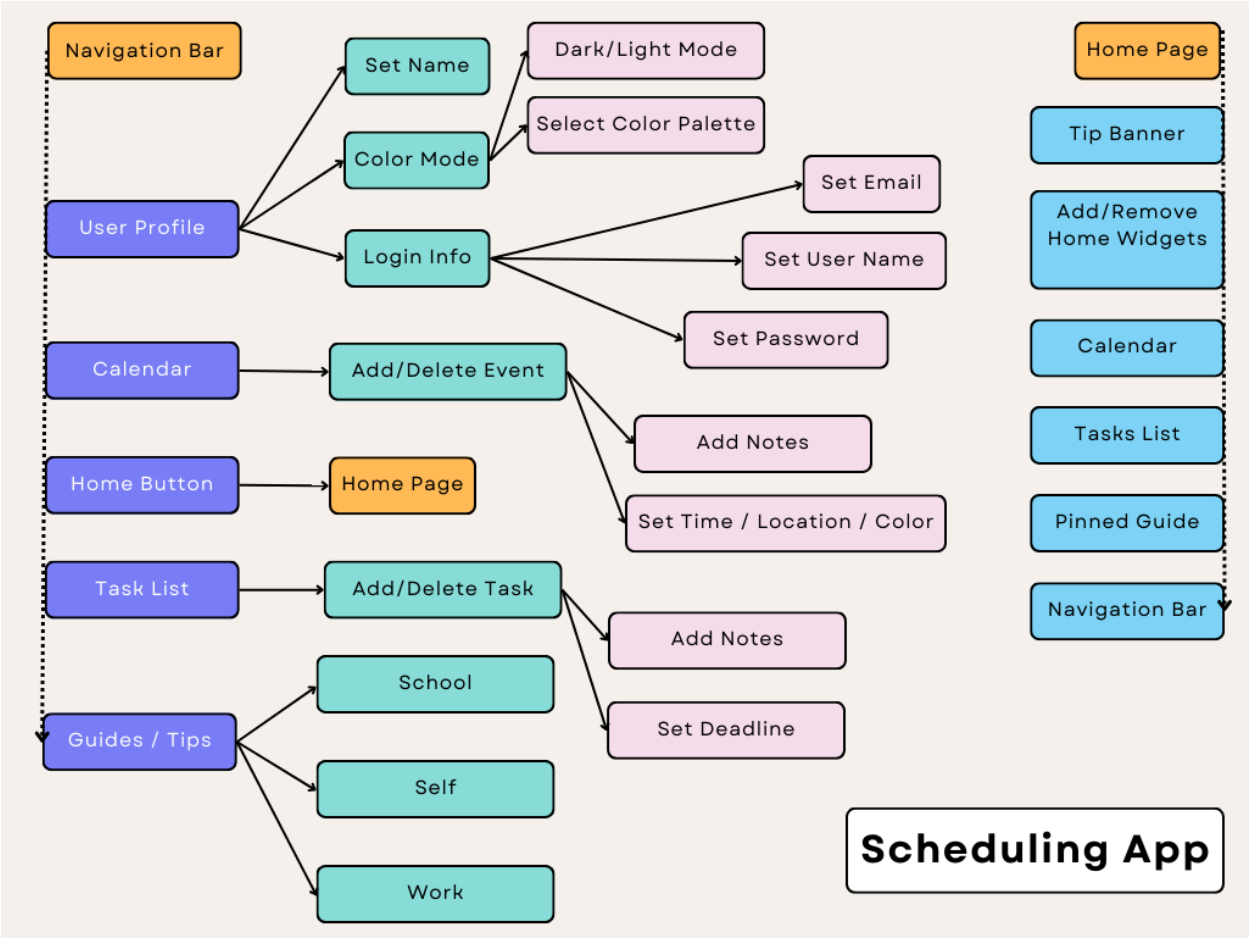
Our testing follows a criterion, risk assessment, resource allocation, and test deliverables for each phase of testing. The completion criteria for the application involve achieving a 90-95% success rate in user usage, ensuring fully functioning software, and promptly addressing any accessibility concerns identified during user testing. During this phase of user testing, our team had multiple different users test out our AdobeXD prototype of Hive Help and much information on how to improve and proceed forward was brought to light. In the next semester, as we convert our prototype into a fully functioning application, we will need to review what failed us in the four stages of development this semester and make changes as needed.

For the development of this app, we will be using the relational database management system MySQL to create a database of educational guide content. We will then implement Express to build the back-end framework of routes. NodeJS will be utilized to create the back-end runtime environment. We will use React to build the user interface. We plan to develop this app to be compatible with Apple iOS and Android in order to make it as accessible as possible.

The documents that we have supplied ourselves with throughout this semester will contribute greatly to the next phase of development. The visual design guide and style guide will provide our group with set parameters in terms of the look of our application. By setting this standard and boundary early in the process, we will not need to scramble to make those design choices in the later stages of production. The development of the testing plan and strategy guide will assist us in the technical, code side of our production and create goals for our team to hit.

The document containing our page tables is key in the coding of our app as well considering it aids us in having to brainstorm content as we enter it in. In short, the collection of all of this data during this semester's efforts will inform every single decision we make moving forward and provides a well-built foundation for which "Hive Help" will stand on.

Information Architecture



➔ Homepage

- ◆ Banner (Top Screen)
  - Randomized Tip of the Day
    - Up to 3 featured
- ◆ “+” Add/Remove Home Widgets
  - Show/Hide Calendar
  - Show/Hide Guide
    - Selects from Pinned Guide
  - Show/Hide Tasks List

- Calendar
- Tasks Lists
- Guides
    - Pinned Guide
        - Only one guide can be pinned at a time
        - If the user hasn't pinned a guide, it will be replaced by a second suggested guide
    - Suggested Guide
        - Links to a randomly chosen guide page, changes when the page is loaded/refreshed
- Nav Bar (Bottom Screen)
    - Guides
        - School Guides
            - Search bar that displays guides that feature the inputted search terms
            - Guides
        - Work Guides
        - Personal Guides
    - User Profile
        - Edit First Name
        - Edit Last Name
        - Edit Email
        - Edit Password

        - Set Color Mode
    - Calendar
        - Add/Delete Calendar Events
            - Set Time
            - Set Location
            - Set Color
            - Add Notes
    - Tasks List
        - Add/Delete Tasks
            - Add Notes
            - Set Deadline (optional)
        - Check off Tasks
    - Home Button

Marketing Plan

Audience: College Students (Ages 18-25)

1. Demographics:
    a. Age: 18–25 years old
    b. Gender: Any gender identity
    c. Location: Primarily residing in or around college campuses
    d. Education: Enrolled in undergraduate/postgraduate programs at universities and colleges
    e. Income: Varied, with some relying on part-time jobs or financial aid

Our media strategy will focus on reaching our audience through digital marketing (social media), influencer marketing, and brand sponsorships. With digital marketing, we will make our app available to users on the Apple App Store or Google Play Store, as well as invest in ad campaigns on various social media platforms (e.g., TikTok, Instagram, Facebook, etc.). For influencer marketing, we will reach out to content creators with platforms related to our app's goals, to advertise our app alongside their own content. We will also focus on brand sponsorships, by featuring exclusive lifestyle tip sections dedicated to another company, in return for them endorsing our app. At this time, we don't plan on integrating our application with social media, besides featuring ad campaigns on various social media sites.

Hive Help Styleguide

Logos

hive help — Light Mode

hive help — Dark Mode

App Icon

App Icon sized 1024px by 1024px.

Identifiers

Primary Color Palette

#F6BD60 #F5CAC3 #F7EDE2 #F28482 #84A59D

#F1E5D2 #382A14 #E1DAD1 #F7F1E7 #C5A36F

Secondary Color Palette

#F6BD60 #F5CAC3 #F7EDE2 #F28482 #84A59D

#AF7029 #362612 #2C1F10 #FECA81 #4C3313

Typography

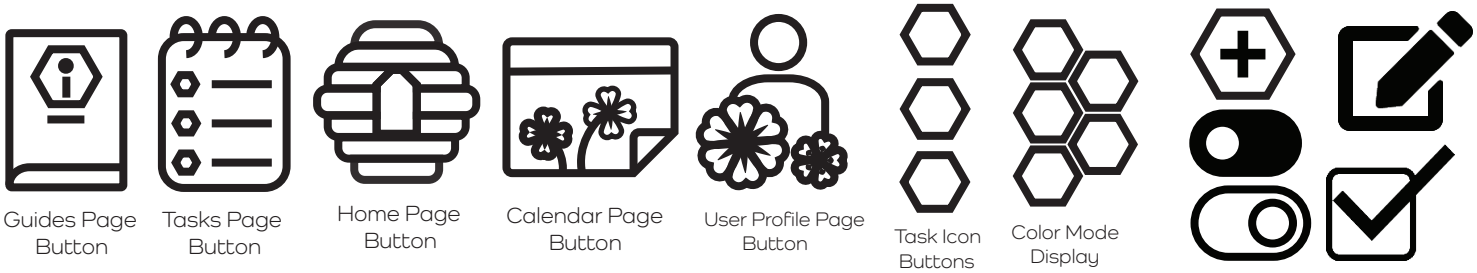
Typography will be using the font "Como" from Adobe Font Libraries.

Aa — **ExtraBold** — Extrabold will be used for our <H1> tagged type. (Titles, Greetings)

Aa — **SemiBold** — Semibold will be used for our <H2> tagged type. (Subtitles)

Aa — **Regular** — Regular will be used for our <p> tagged type. (Button titles, paragraph text,)

Icons & Buttons



Guides Page Button · Tasks Page Button · Home Page Button · Calendar Page Button · User Profile Page Button · Task Icon Buttons · Color Mode Display

Forms & Inputs

User Registration

Name *
First | Last

Email *
Email Address

Confirm Email *
Email Address

Password *
Password

Confirm Password *
Password

Accept Terms and agreements.

Register

User Sign In

Sign In
Email
Password
Forgot Password?
Sign In

Don't Have an Account? Register Here!

Create Event

Event Title
Enter Title

Notes
Enter Notes

Reminder
Date: MM/DD/YYYY
Time: HH:MM:XM

Banner Color

Back | Confirm

Forgot Password?

Email
Send Email

You will receive an email in around 5 minutes.

Change Password

Enter New Password
Confirm New Password
Change Password

Task Name
Enter Task Name

Task Description
Enter Description

Due Date
Enter Due Date

Back | Confirm

Accessibility

Hive Help Visual Design Guide

Logos

Logos provided are a dark and light mode logo based on user settings. We wanted a curvy bee themed logo that is based on our hive mind theme. On the left, we have a light mode logo and then on the right we have a dark mode logo.

What to Do with Hive Help Logo

✔ Use Simple Identifiers
✔ Use Readable Font
✔ Use Brand Color Palettes
✔ Meet Branding Needs

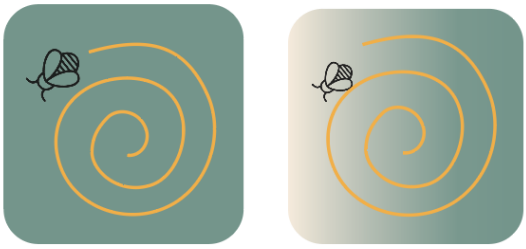
What Not to Do with Hive Help Logo

✘ Use Complex Fonts
✘ Unrecognizable Colors
✘ Add Unnecessary Images or Text
✘ Skew from Branding Needs

Identifiers

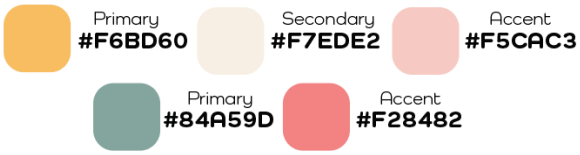
These are hive help identifiers and will be used as a secondary logo that does not have any text. You can find these identifiers within the app and are an extra part of Hive Help's branding.

App Icons



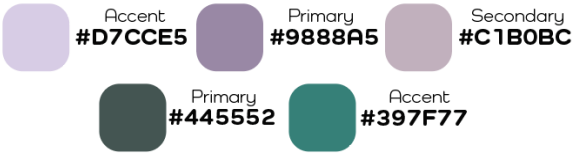
Our app icon is sized to the correct pixel size provided by Apple and Android requirements. The size is 1024px by 1024px and the app icon file is located in the ZIP folder.

Primary Color Palette



Primary Color palette is supposed to be welcoming and reminding of colors people would think of when thinking of bees and honey. We have listed the Primary, Secondary and Accent Colors.

Secondary Color Palette

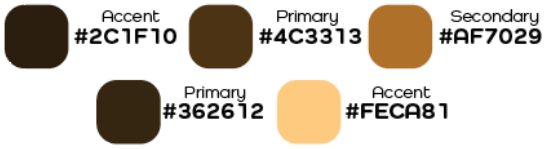


The Secondary color palette is supposed to be a darker version of the original palette. It offers users a second choice of what they want to see when they use the application.

Primary Color Palette #2

Primary Color palette is supposed to be welcoming and reminding of colors people would think of when thinking of bees and honey. We have listed the Primary, Secondary and Accent Colors.

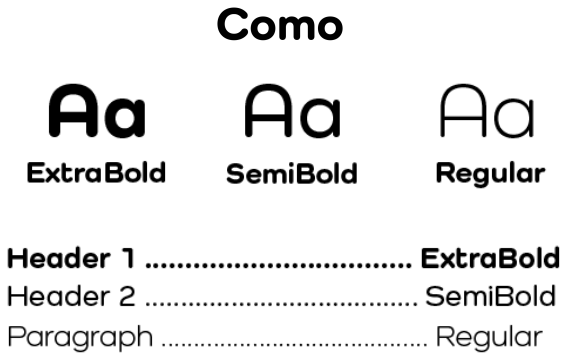
Secondary Color Palette #2



The Secondary color palette is supposed to be a darker version of the original palette. It offers users a second choice of what they want to see when they use the application.

- All color palettes can use different hues and saturations to create depth and variation in design. They should still follow the main primary, secondary and accent colors.

Typography



Our typography uses the font “Como” which is an Adobe Font. We have decided to go with a single font and as shown to the side each style and their use.

<H1> Tags will use Extrabold, <H2> will use Semibold and <p> text will use the regular styling.

Navigation Buttons & Icons

These are the Nav button icons for our navigation bar that will always be visible to users.

They are labeled to the side and are planned to be in the order they appear. Again, we wanted to stay consistent with the bee, flower and honey theme.

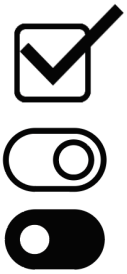
Honeycomb Check Boxes

These honeycomb check boxes will be used for marking tasks complete or incomplete. If they are complete there will be a checkmark. If the task is incomplete the honeycomb will be empty.

Color Palette Display

The user will be able to select between a light and dark mode. These color palettes for the respective mode will be displayed in this honeycomb layout.

Navigation Buttons



#1. The reminder icon button for when users create new calendar events.

#2. This will be the button for the user to switch between dark and light mode. When the user is switched to light mode the button will be unfilled. When the user is switched to dark mode the button will be filled.

#3. The edit profile button will allow users to edit their personal information.

#4. The button to open up certain menus, and add tasks, events and guides.

Accessibility

We wanted to make our text as readable as possible and that is why we used a dark color on our light mode palette and a light color on our dark mode palette.

These are just some examples on what it would look like in our application.

User Sign In

This is our sign in form, where the user's info will be stored and linked to their account.

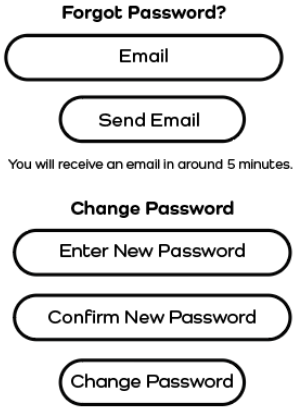
We also have a section for forgetting their password and registering for an account if they don't have one.

User Registration

This is the user registration form. The user will be able to enter their name, email and create a password. They have to accept our application terms and agreements before registering.

The username and password will be case sensitive and meet certain requirements.

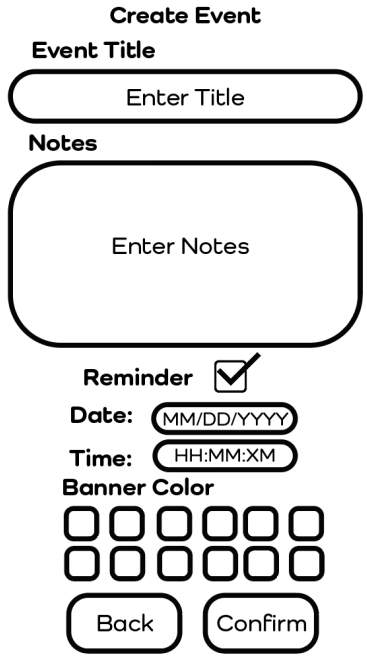
Forgot Password Form



These are the two forms that will be used to help a user reset their password if they have forgotten it.

They will have to enter their email to receive a link and from there they will be able to set a new case sensitive password.

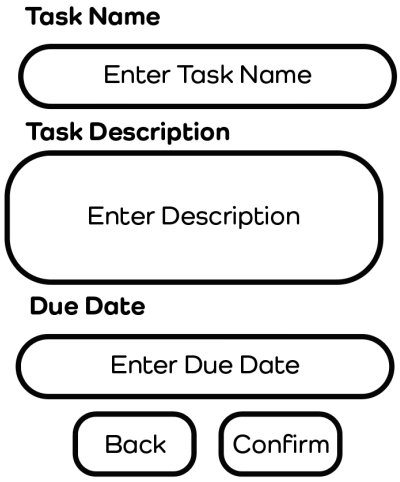
User Create Calendar Event



This is the form for users to create a new calendar event. They will be able to enter an event title, event notes and mark if they need a reminder or not.

The user can also decide if they want the reminder to happen at a certain date or time. The banner color will be what color is displayed on the calendar and then the user can go back or confirm the event.

User Create New Task



The create new task form is for users to create new tasks for their task lists.

They can enter a task name, description and due date. This is similar to the calendar form but a little more simplified.

They can then go back or confirm.

Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

Hive Help Front-End Development Phase

Executive Summary

This phase of our project focused on the front-end development of our mobile application. Based on our prototypes in AdobeXD, we started developing our app using React Native Expo. We worked hard in this phase to make a visually appealing application for our users, and we look forward to the next phases where we can flesh out our application even more. In this phase, we divided the pages/screens amongst ourselves, and recreated our prototype's design to the best of our ability. We decided to make some quality-of-life changes, like splitting up the profile and settings screens, and how the guides are viewed. Some changes were made because we were very ambitious in our initial designs, and other changes were made due to making it easier for the user. We are very happy with our end result, and look forward to developing it even further.

Documentation

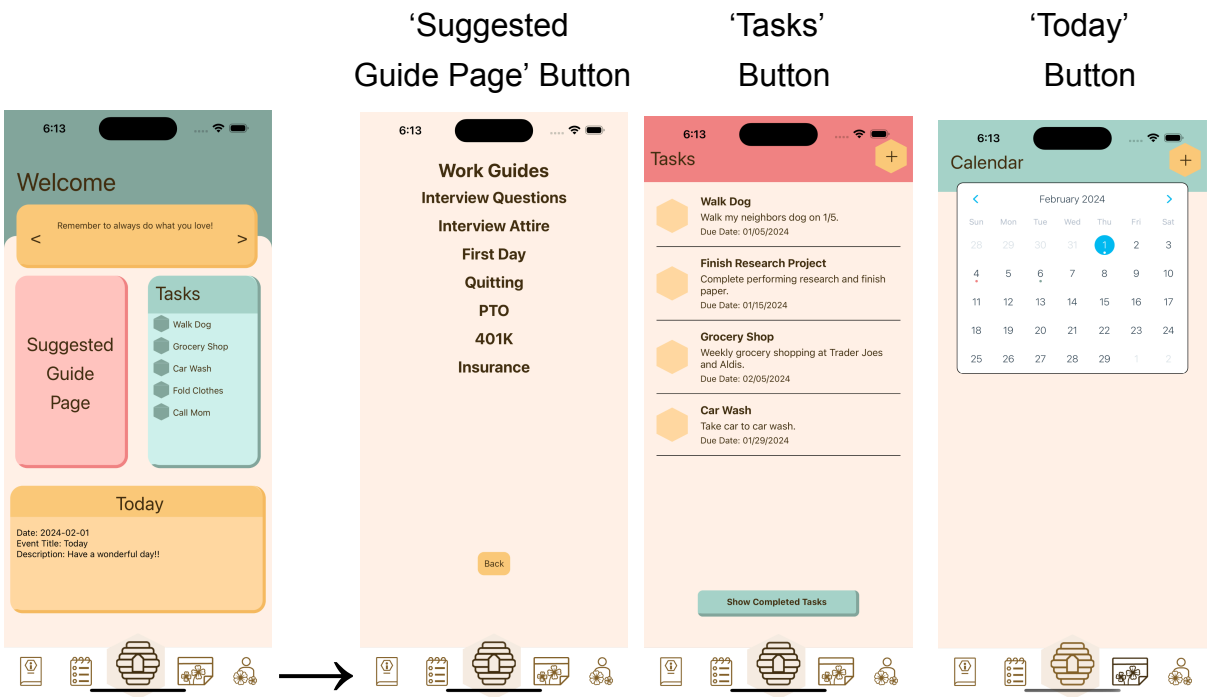
Home Screen

Design assets:

- 'stylesheet.create' was imported to assign different color codes to each component depending on which color scheme the user selected in the Profile Screen.

User interface design:

- The homepage allows for users to see important information from the other screens in the app.
- The top most box utilizes UseState by pressing the carrots to right or left of display text in the box in order to change the display to the alternative display text



Technical implementation:

- The Home Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity'', and 'useState'. The main container for this page is a 'View' component that holds all the other components. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the task screen, work guides screen, and calendar screen. UseState is utilized by pressing the carrots to the right or left of display text in the up most box in order to change the display to the alternative display text. TaskList.Json is imported and the JSON is set to the variable 'tasks' while utilizing useState and displayed to the screen using map() in order to get the name of each task and display it in the 'Tasks' button. New Date() is used to assign currentDate with the current date

and then converted to a string. eventDetailsJSON is imported from eventDetails.json in order to display the date, title and description of the eventDetails in the 'Today' button.

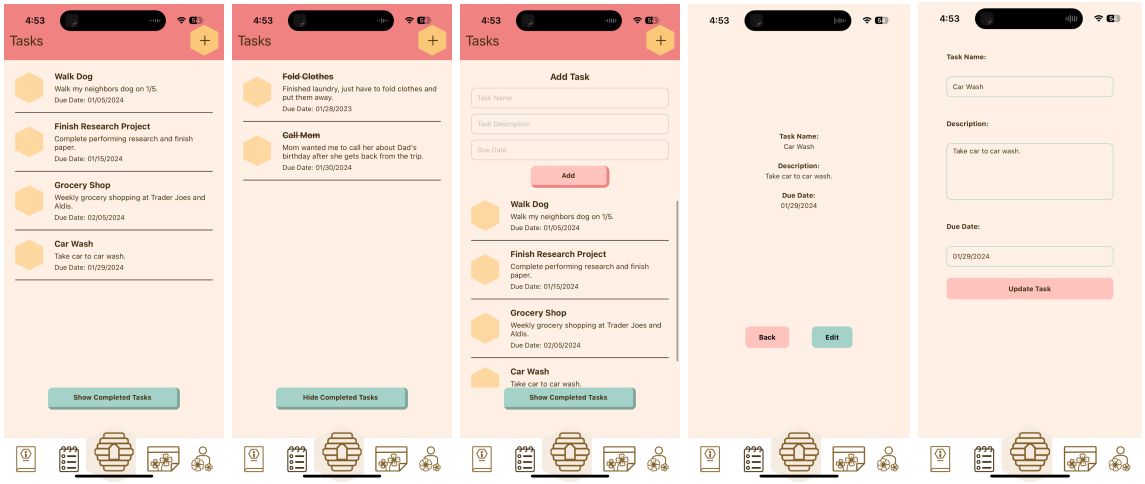
Tasks

Design Assets

- The design for the 'TaskScreen' page and its connected screens was changed in order to better work with our skills and abilities. We came to the consensus that it might even look better than our hi-fi prototype for simplicity and ease of use. The 'stylesheet.create' was used to control the design of the task screens.
- * For reference when looking at my screenshots, I have the iPhone 15pro and my font in settings on the smallest option. Results may vary.
    - I used 'AntDesign' which is a react native icon library and it helped me through getting an accurate hexagon shape. This also included a plus icon in the middle of the hexagon specifically for the add button. The checkboxes are identifiable by being a yellow hexagon shape.
    - We followed our color themes: 'lightA,' 'lightB,' 'darkA' and 'darkB.'
    - There are a few text field input boxes in the task portion of the app like on the add task screen and edit task screen.
    - I made the use of normal buttons as well that are styled to match the applications theme settings. This includes color changes and shadows.
    - My screenshots below are using the 'lightA' color mode.
    - I do plan on tweaking style a bit if we do decide to stay true to the hi-fi prototype I may implement a grid layout.
    - Our font size is set in our styles but depending on the size of your phone text it can alter the size of how things render.

User Interface

- These are my main screens for user interface. There are three main screens, 'TasksScreen,' 'TaskDetailsScreen' and 'EditTasks.' Those were the main focus for the time we had for the front - end side of things.
    - The main 'TasksScreen' displays the uncompleted tasks, these are hard coded for now until we start storing data in the backend and focus on updating the data to render the changes. The tasks are listed and separated by a bottom border width. The yellow hexagon next to the tasks can be clicked to move the task to complete.
    - From the 'TasksScreen' you are able to press the hexagon button that opens an add task field. This allows users to input a task name, task description and task due date. Followed by an add button. It also includes a 'show completed tasks' button which will render the completed tasks which you can hide with the 'hide completed tasks' button.
    - The 'EditTasks' screen gives you the option to update the task and also incorporates text fields for the user to input updates to their tasks. This is followed by an update task button.
    - The 'TaskDetailsScreen' displays the singular task and the buttons to go back to the main page or to edit and update the task.
    - For now until we get to the back - end side of things the 'HomeScreen' uses a json file to hard code in values for what we want it to look like at the end of development.
    - The goal was for the user to have a simple scrolling list of their tasks and a way to filter through what they have done and what they have to do.



Technical Implementation

- I wanted to use a similar structuring to a simple task list design we used in mobile development. This includes using a few different libraries, states and components to pull it all together.
    - Some of my imports included React hooks like 'useState' and 'useEffect.' The 'useEffect' populates the initial tasks that are hardcoded for ease of completing the front-end side of things. The 'useState' manages the functional components and updates the value. I also used 'Ant Design' and View, Text, StyleSheet, FlatList, TouchableOpacity, TextInput. Those are all pretty standard for rendering your content. We imported 'Theme' from our theme file to stay consistent with our styling as far as colors, text sizes and buttons. I also imported 'useNavigation' so that the user can navigate between task screens.
    - My main component 'TaskScreen' serves as the main manager for what happens with my tasks. It handles my navigation, state management, event handlers, rendering and more that I will describe below.
    - I have a 'useEffect' function that contains an array of predefined json data so that it can be rendered on the task page. My json data is in an array and contains name, description, due date and completed boolean.

- I used 'FlatList' to render my data onto the page. This alo contains the organization for the styles based on their respective property to be styled within the stylesheet. Return is also used and helps when organizing what is styled with what.
- My 'StyleSheet.create' contains all my styling for the container, buttons, text, etc. It is a little chaotic and I will look into condensing as needed in the future.
- For checking to make sure input fields aren't empty strings I implemented 'trim().' This is helpful in the add task function and the edit task update function.
- I like to use 'Touchable Opacity' so that you don't necessarily have to get a bunch of different screens to work together. The visual change on the same page should be plenty for the user to easily complete the task at hand.
- The use of 'route.params' has also been helpful in making sure my data is passed correctly from screen to screen. This way the tasks don't get mixed up.

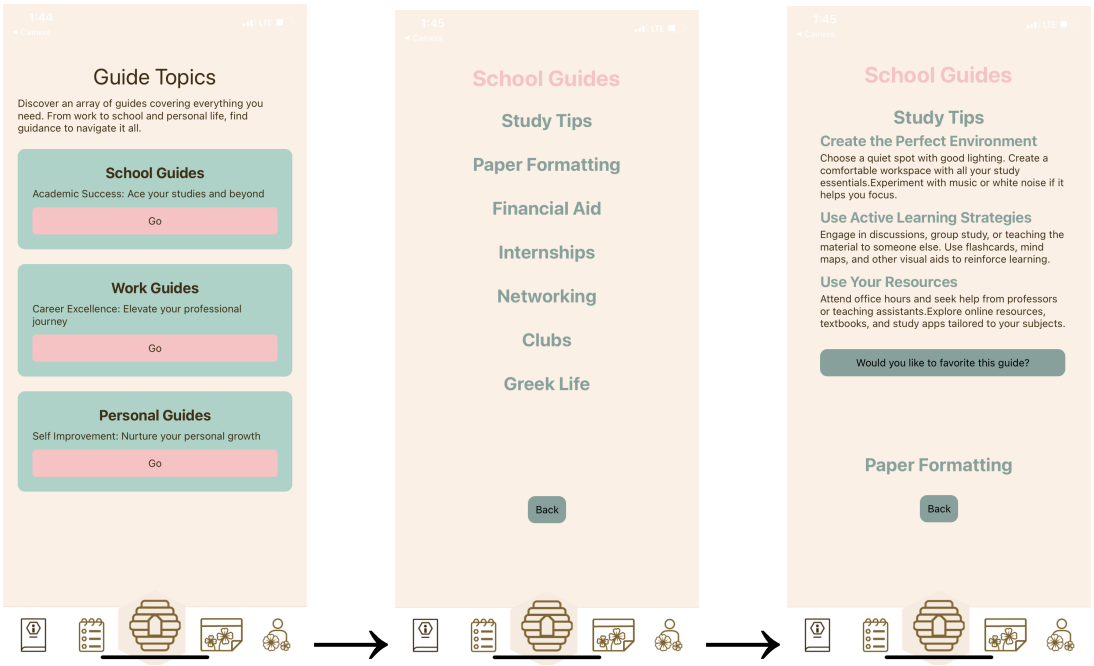
Guides

Design assets:

- During development, we had to make some revisions to the design. Instead of using the Como font, we decided to use a different sans serif font that will be easier for the user to read. Also revisions within our color scheming. The colors that we have are as close to the original palette as possible for our primary color scheme.

User interface:

- The main guide screen design has changed in accordance with accessibility and ease of use. The screen displays the cards for "School Guides", "Work Guides", and "Personal Guides". Each has a small descriptor of the type of content within each genre of guide. Each card has a button labeled "Go" that, when clicked, takes the user to the corresponding guide topics page.
- The guide topics, when clicked, will display the information about that topic separated into smaller headings. The display is an accordion animation/ dropdown that can be scrolled through. Once a new topic is clicked, any opened topic will collapse and the new one will open.
- Buttons:
    - Back Button- this button has functionality that will take the user back to the main guides page
    - "Would you like to favorite this guide?" Button - This button will call the specific guide topic that is clicked and display it in the "Favorite Guides" section of the User Profile page. (further back-end functionality needed)



- There were significant changes made to the guides user interface. In our prototype, the user would have needed to navigate to different screens for each topic within a guide genre. Instead, we've implemented the dropdowns to cut down on the number of screens we would have had to make during the front-end development.

- Technical implementation:
    - The structure of the main guides screen was created by using a functional component to map through the "guideCards" array and display them as TouchableOpacity cards. For the "Go" buttons, TouchableOpacity is also used and "useNavigation" from react-native/navigation was used to create the routing between the main guides screen and each topic's corresponding content screen.
    - School.js, Work.js, and Personal.js all share the same structure and functionality. Each of these screens is linked to a different set of data stored in corresponding JavaScript files (SchoolGuideData.js, WorkGuideData.js, and PersonalGuideData.js). The data is structured to include a title for each guide's subtopics, the subheadings, and then their content
    - Using FlatList and TouchableOpacity, the list of the subtopics is rendered on the page and once clicked opens up to all the content within that guide.
    - We used useState hooks and event handlers to define and change the states of the clickables and buttons
    - In each file, there is a renderSections function that is set to display the dropdown content since in the JSON data structures the subheadings and content are nested within a sections array.

Calendar

User Interface

- The design for the 'CalendarScreen' page, much like the Tasks, was modified to function within my capabilities and understanding of React Native as well as for simple usage. Instead of allowing for multiple screens to be opened from the main 'CalendarScreen', I utilized modals to open up both the Event Details and the Add Event sections. The main screen displays a calendar imported from react-native-calendars and is displayed underneath the heading 'Calendar' and the 'Add Event' button.

- Underneath the calendar is a blank section that is space for the user to open up the event details. Within the event details is a heading of the event title as well as the Notes, Date, and Time for the event that the user opens. Upon display of these details, are also 'Back' and 'Edit' buttons. Once the functionality is completed, the user will be able to edit the events, currently only the 'Back' button allows the user to return to the main calendar while the 'Edit' button remains static.
    -
- When the user opens the 'Create Event' modal, the modal appears as its own screen within the 'CalendarScreen'. Displayed is a 'Create Event' heading, as well as text labeling each of the text inputs. Above each input is labels for the Event Title, Notes, Date, Time and Banner Color. The inputs are formatted as a blank white box and the banner colors are six various pastel colors formatted as circular. Below all of these inputs are the 'Back' and 'Create' buttons.

Design assets

- During the construction of this portion of the application, much had to be modified from the prototype to React Native. We used the stylesheet.create for building

the design of the CalendarScreen. An external ThemeProvider.js was imported to the screen to be able to call the color themes. This portion of the application does use styling very similarly to the TasksScreen for cohesion.

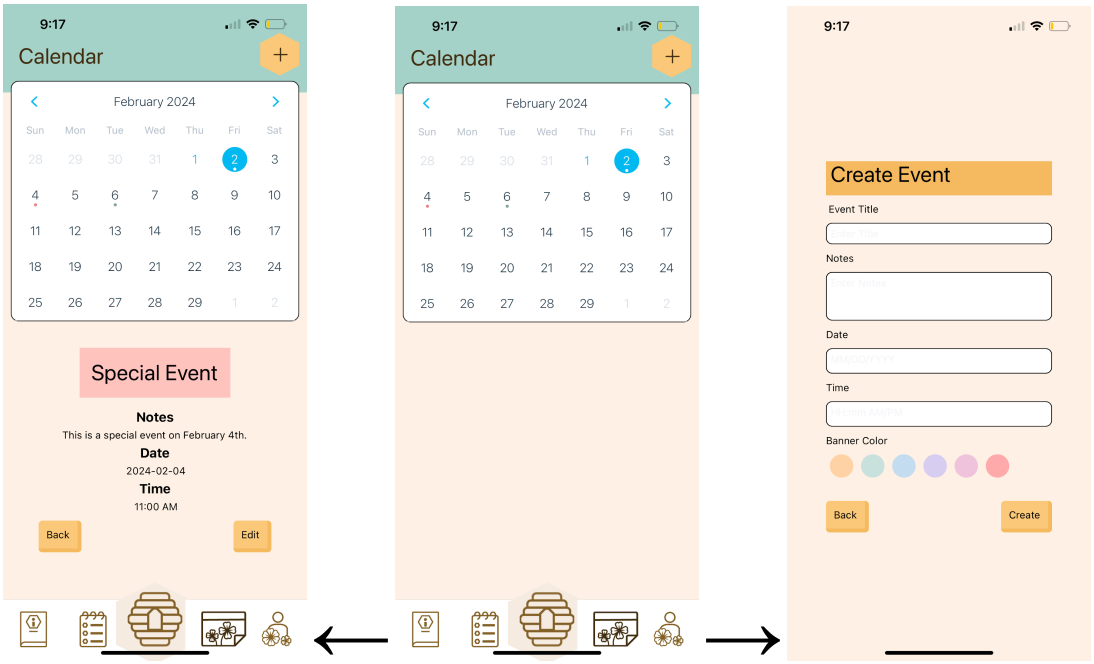
- The react native library 'AntDesign' was used when creating the hexagon shape for the Add Event button.
- All four color themes are functioning and are applied throughout this screen when the theme is changed through the settings.
- The font was not able to be imported during this phase, and will be implemented into the app soon. Currently, the font size is set to the size on the user's device.
- Pastel colors are set as color options for the calendar event banner color. These can be viewed on the Add Event modal, and will eventually be implemented on the Edit Event screen.
- Technical implementation: A description of the technical implementation of the user interface, including details on the HTML, CSS, and JavaScript code used

Technical Implementation

- The structuring of the Calendar screen is based around the Calendar imported in form react-native-calendars.
- The calendar functionality is implemented using the 'react-native-calendars' library. It allows for events to be marked on the calendar and for event details to be displayed back to the user.
- For state management, I used the React hooks, 'useState' and 'useEffect'. The variables include 'showModal' to handle opening and closing the event details as well as the add event modal.
- The event data management has begun its construction, but is not yet fully functioning. The useEffect hook will be used to add new events via our addEvent function. Currently, the markedDates and eventDetailsJSON contain preset events as well as the current date as an event, but they are not able to be

updated yet. We plan to repair this issue and create the functionality during our back-end development phase.

- There are multiple user inputs for adding the event title, notes, date, time and banner color, but they are yet to be functional. We will be implementing that as well during our back-end phase.
- The buttons utilize the TouchableOpacity and are used to close modals upon press.



Profile Screen

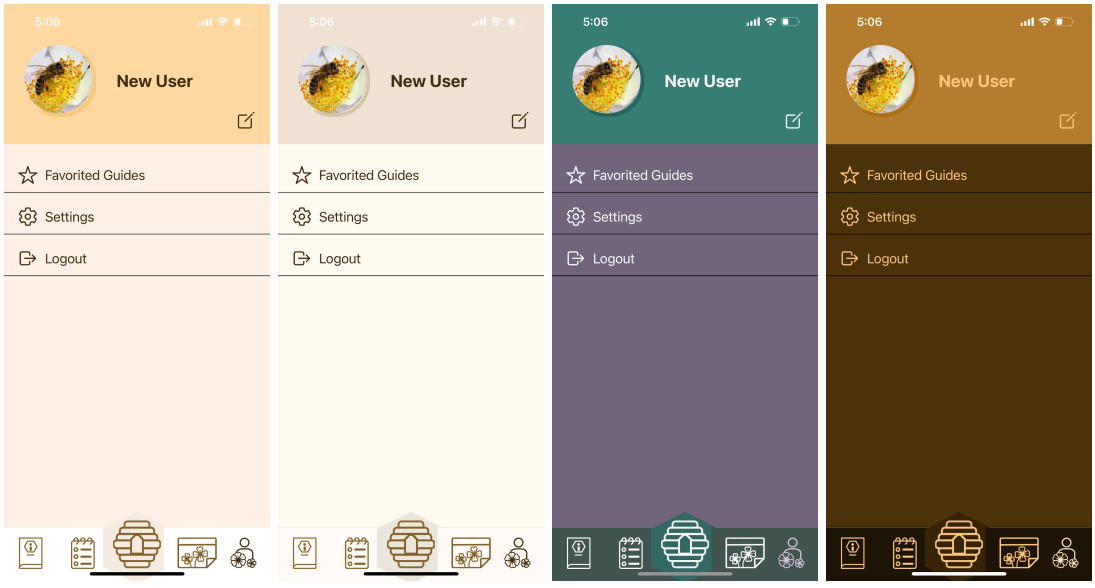
Design assets

- A stock image of a bee (./assets/bee_icon.jpg), is used as a placeholder profile picture.
- A few icons are used from the React Ionicons component, specifically:
  - star-outline
  - settings-outline
  - log-out-outline

- create-outline


User design interface:

- The design of the profile page has been changed from what it originally was in the high-fidelity prototype. These changes were made to give the user a more visually appealing experience, as well as allow them to navigate more seamlessly through the pages. The user is greeted by a simple profile page, displaying their profile picture, saved name, and a small vertical navigation menu below the header. This menu includes a button to view the guides that the user has favorited, a button to navigate to the settings menu, and a button that logs the user out and takes them back to the login page. The "edit" icon on the bottom right of the profile header is for the user to edit their name. Clicking this lets the user edit their name on the same page, and either save their new inputted name, or cancel the interaction.



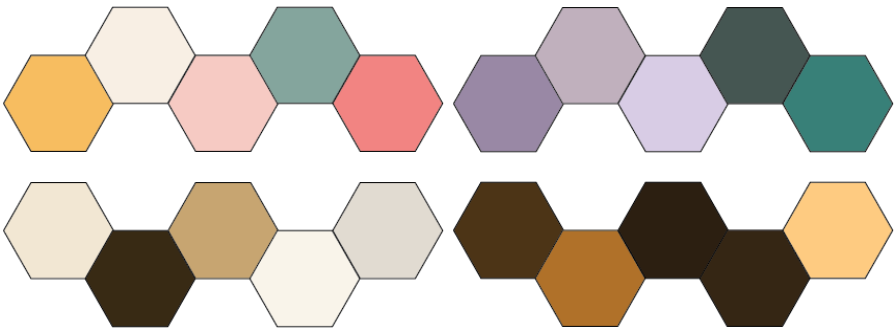
Technical implementation:

- The profile page is built using various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity','TextInput', and 'Button'. The main container for this page is a 'View' component that holds all the other components. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook. For the user to edit their name, we used the 'useState' React hook, to set a default name, and apply the changes when a user updates their name.

Settings

Design assets:

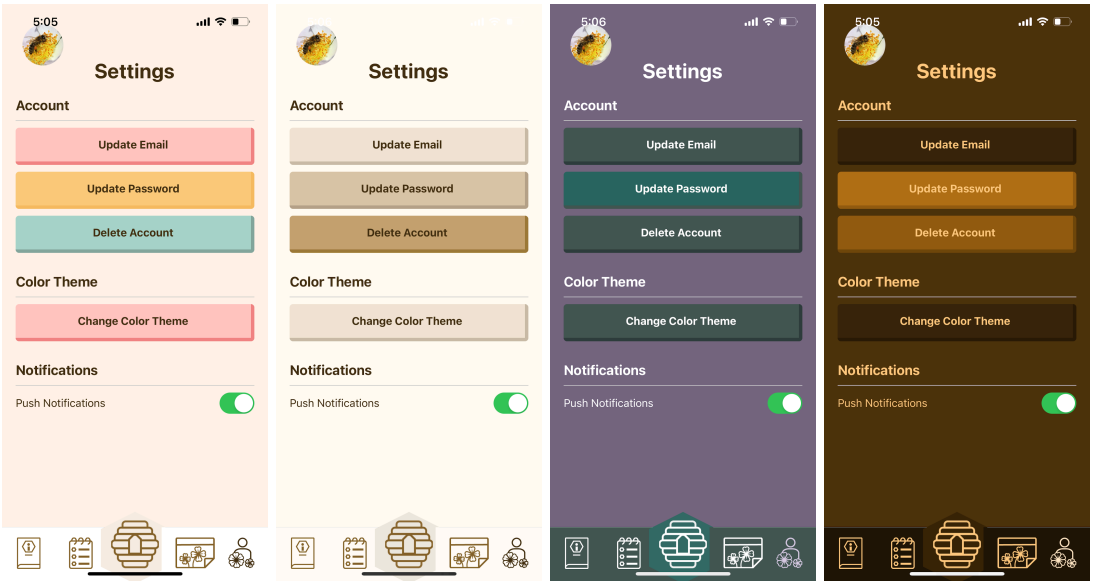
- Four images are used to display the available color palettes to choose from, (light_mode_1.png, light_mode_2.png, dark_mode_1.png, dark_mode_2.png).
- The example profile picture from the profile page is also displayed on this page.



User interface design:

- The settings page wasn't one of the screens in the original prototype, but was created to make the profile its own thing. The purpose of this screen is to separate the app/account settings options into one spot where they

can all be accessed at once. This page just consists of headings and corresponding buttons to the headings, with sections for "Account", "Color Theme", and "Notifications". Under "Account", the user can update their email address associated with their account, update their password, and delete their account if they wish to do so. Under "Color Theme", the user can click the button to view the two available swatches for light mode and the two available for dark mode. Under "Notifications", the user can toggle whether they want push notifications on or not. The profile picture in the top left corner takes the user back to the profile page.



Technical implementation:

- Just as the profile page, the settings page is also built using various React-Native components, such as 'View', 'Text', 'Modal', and 'Switch'. The main container for this page is a 'View' component that holds all the other components, and each of the modals have a render function. The styling of this page uses the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook. The 'useState' hook is used frequently

on this page, to determine which theme the user has selected, as well as handling the popup displays of the modals being shown/hidden.

Sign In Screen

Design Assets

- SignInBackground
- Como font

User interface design:

- The top most box with the placeholder text 'Enter Email' allows users to enter text
- The second most box with the placeholder text 'Enter Password' allows users to enter text
- The button titled 'Sign In' navigates users to the Home Screen
- The button titled 'Create Account' navigated users to the Create Account Screen

Sign In' Button 'Click Here' Button

Technical implementation:

- The Sign In Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity'', 'Dimensions' and 'TextInput'. The main container for this page is a 'View' component that holds all the other components. 'Dimension' is used to find the dimensions of the user's screen in order to correctly size the background image to the screen. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the Home Screen and the Create Account Screen.

Create Account Screen

Design assets:

A list of the design assets (such as images, icons, and fonts) used in the mobile application user interface

- CreateAccountBackground
- Como font

User interface design:

- The top most left box under 'Name*' allows user to input text
- The top most right box under 'Name*'' allows user to input text
- The box under 'Email* allows user to input text
- The box under 'Confirm Email*' allows user to input text
- The box under 'Create Password*' allows user to input text
- The box under 'Confirm Password*' allows user to input text
- The button containing 'Sign In' text navigates to the Home Screen
- The button containing 'Click Here' text navigates to the Create Account Screen

'Sign in' Button 'Click Here' Button

Technical implementation:

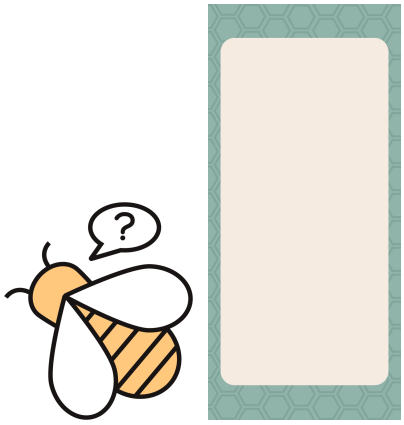
- The Create Account Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity'', 'Dimensions' and 'TextInput'. The main container for this page is a 'View' component that holds all the other components. 'Dimension' is used to find the dimensions of the user's screen in order to correctly size the background image to the screen. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the Home Screen and the Sign In Screen.

Forgot Password Screen

Design assets:

- bee-icon
- greenBackground
- Como font

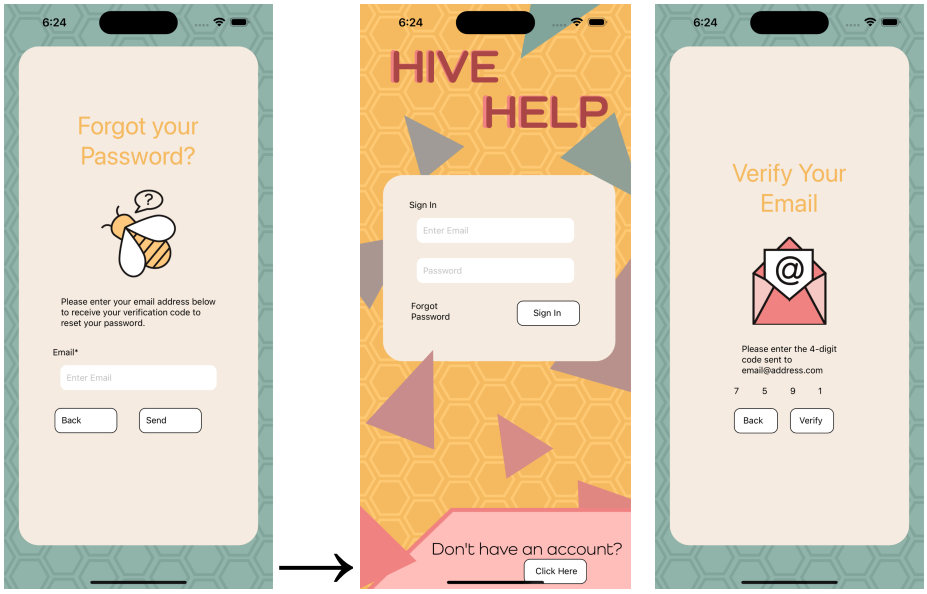
Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein



User interface design:

- The box containing the placeholder text 'Enter Email' allows user input text
- The button containing the text 'Back' navigates to the Sign In Screen
- The button containing the text 'Send' navigates to the Verify Email Screen

Back' Button 'Send' Button



Technical implementation:

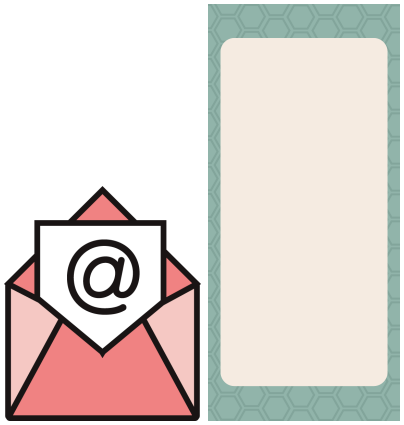
- The Forgot Password Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity'', 'Dimensions' and 'TextInput'.

The main container for this page is a 'View' component that holds all the other components. 'Dimension' is used to find the dimensions of the user's screen in order to correctly size the background image to the screen. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the Sign In Screen and the Verify Email Screen.

Verify Email Screen

Design assets:

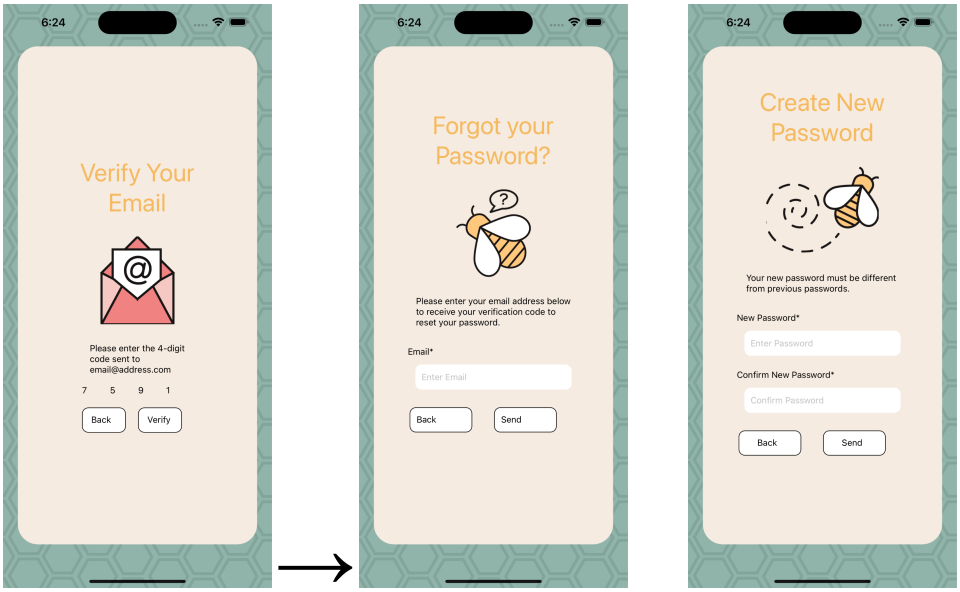
- mail-icon
- greenBackground



User interface design:

- The button containing the text 'Back' navigates to the Forgot Password Screen
- The button containing the text 'Verify' navigates to the Create New Password Screen

'Back' button 'Verify' Button



Technical implementation:

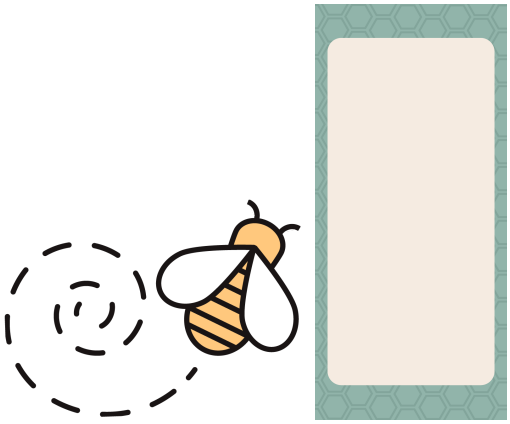
- The Verify Email Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity' and 'Dimensions'. The main container for this page is a 'View' component that holds all the other components. 'Dimension' is used to find the dimensions of the user's screen in order to correctly size the background image to the screen. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the Forgot Password Screen and the Create New Password Screen.

Create New Password Screen

Design assets:

- bee-fly-icon
- greenBackground

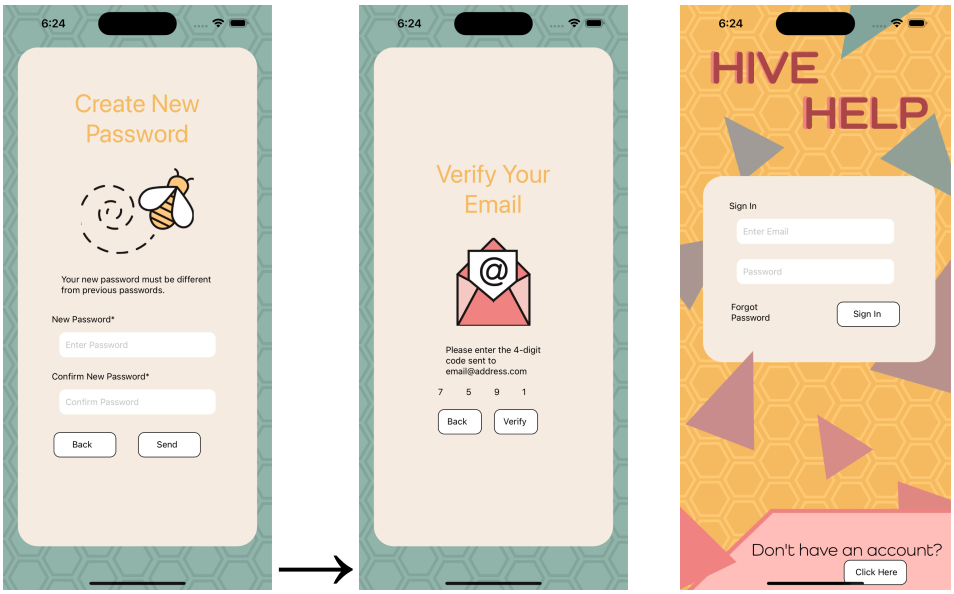
Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein



User interface design:

- The box containing the placeholder text 'Enter Password' allows user to input text
- The box containing the placeholder text 'Confirm Password' allows user to input text
- The button containing the text 'Back' navigates to the Verify Email Screen
- The button containing the text 'Send' navigates to the Sign In Screen

'Back' button 'Send' Button



Technical implementation:

- The Sign In Screen is composed of various React-Native components, such as 'View', 'Text', 'Image', 'TouchableOpacity'', 'Dimensions' and 'TextInput'. The main container for this page is a 'View' component that holds all the other components. 'Dimension' is used to find the dimensions of the user's screen in order to correctly size the background image to the screen. The styling of this page is done in the 'CSS-in-JS' styling technique, using the 'Stylesheet' module provided by React-Native. The navigation on this page is implemented with React Navigation's 'useNavigation' hook and is used to navigate to the Verify Email Screen and Home Screen.

Testing and debugging

Platform

- We have been using VS code along with installing Expo Go on our personal devices which has allowed us to easily see our changes in design and functionality in real time.
    - We decided to create the react native app using the Expo package because it was familiar and easy to use.
    - The updates in real time have proven to be extremely helpful when it comes to styling as we are able to resolve issues fairly quickly.
    - Some of the issues we ran into with expo go was having to use '--tunnel' when we would run npx expo start. Some expo versions were outdated and needed to be updated causing issues or there were issues in our code that needed to be resolved.

    - Since we are working through Github we were having quite a few issues with our branch and merging everyone's content together but again, that was easily fixed through communication.

Debugging

- When we would get console errors or errors in the expo go app we would use resources like chatGpt and stack overflow to try and resolve these issues.
    - A lot of these issues were missing packages that we fixed by installing them in the terminal.
    - Other issues were things we decided would be worth putting energy into when we dive into back end development. These are things like creating/updating tasks and events in the task and calendar screens. We were able to get the task to add but it is not stored as user data it is simply rendered to the screen. And the calendar is able to display hard coded values as well but we were having issues getting the events to be added. Again, we decided to focus on this full force in the next development phase.
    - We had some navigation issues between our screens but we were able to create separate stack navigators that resolved that issue.
    - I think we have all tried out navigating the app ourselves and have had some people tap through it already.
    - Since we are doing a mobile app that relies on functionality we could not just create static pages. This turned out to be a little stressful as we may have underestimated the amount of functionality needed to go along with styling. We have plenty of kinks to work out but for now, the main issues have been resolved into our simple functioning app.

Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

MS6: Hive Help Back-End Development

Executive summary

For this milestone, we focused on creating our database and updating all of our content in the application to be stored on Supabase. Our app is still run on Expo Go which works with our React Native framework. We ran into a few issues deciding which API and database to use but eventually settled with Supabase. One of the main focuses was making sure all of us had access to the database and we wanted ease of use when deciding what to use. With that we divided and conquered into our section of the application and made our respective tables in Supabase. We were mainly focused on getting things to be stored in the database and will work on making client and server work seamlessly together. Now that we have a server up and running we look forward to connecting our sections of the app. Eventually, we managed to get quite a bit done in the time we had and look forward to tweaking the bits and pieces that need it. Overall, our server is up and running and users are able to create an account and even complete some of the tasks in the different sections of the application.

Server-side architecture

A description of the server-side architecture, including details on the programming languages and frameworks used

- The app is being hosted on an external server on Supabase, with a PostgreSQL database. To handle higher amounts of user authentication on our app, we are using a Simple Mail Transfer Protocol (SMTP) on Resend.

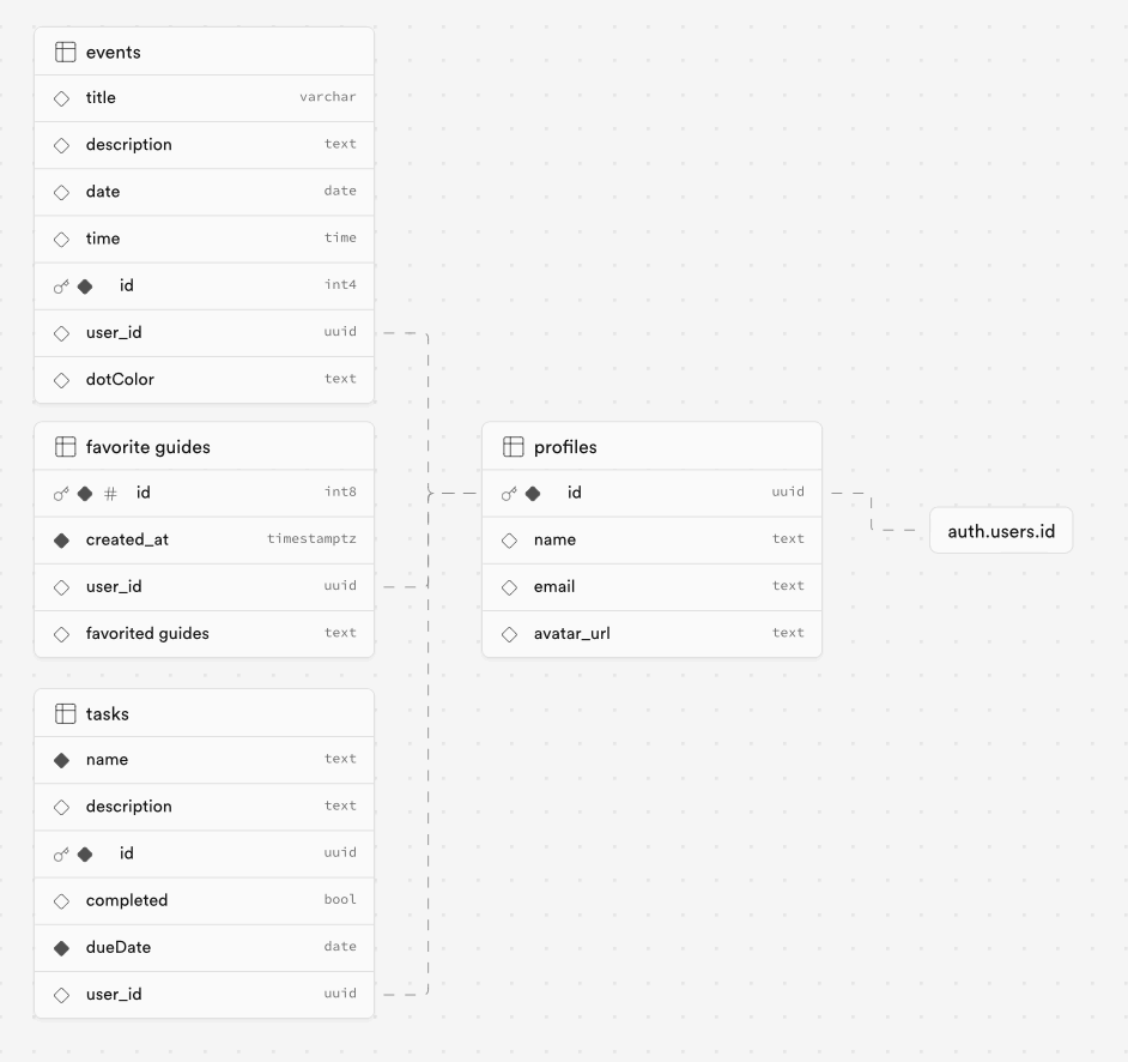
Database integration

A description of the database integration, including details on the database management system used and any data models or schemas

- Account information is stored in the authentication database in order to utilize the password encryption and recovery options provided. Each user is assigned a unique User UID which is referenced in other tables as the foreign key to retrieve data that belongs to the corresponding user.
- The events, favorite guides and tasks tables all use the foreign key to interact with user-specific data.

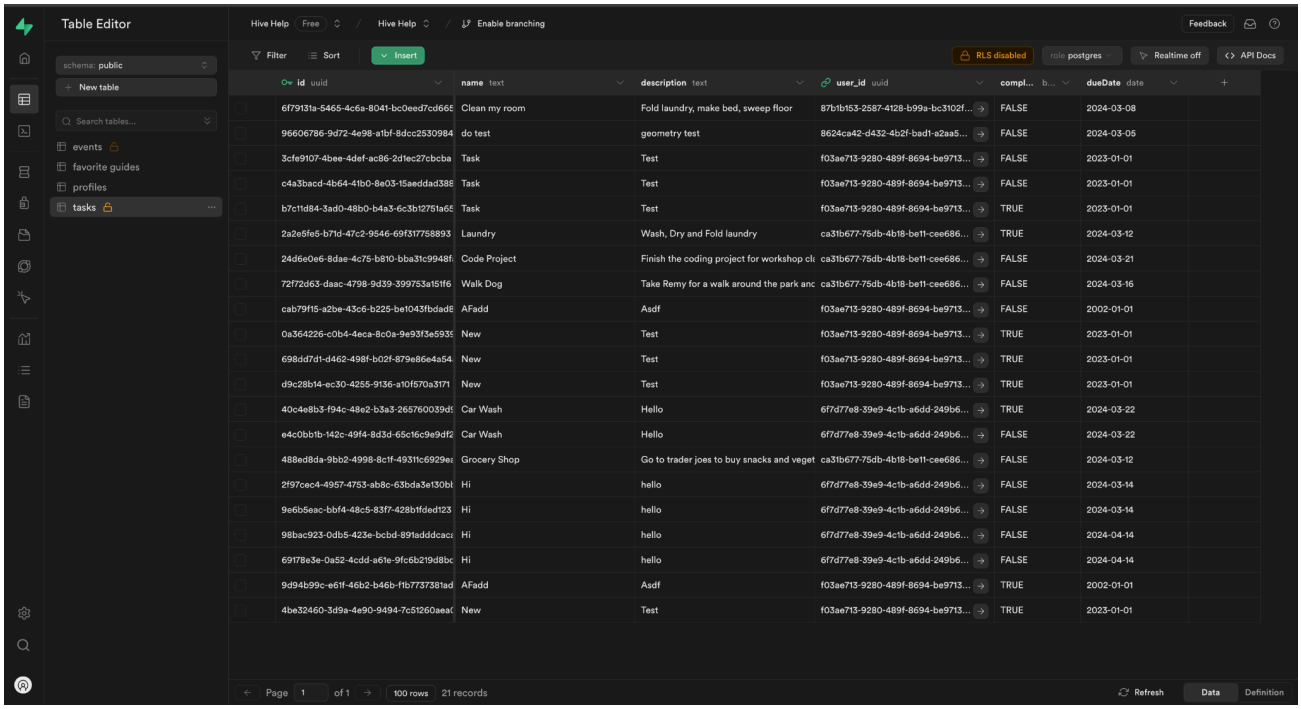
Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

-



Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

Tasks



User Interface / Server-Side Architecture

- The users are able to now add tasks tied to their specific account, this is then stored in the database. Before the tasks were only stored by being hard coded and were deleted once the user logged out or closed the app.
- The table was created based on the attributes of the tasks themselves.
- The table was created by using the sql editor in Supabase, I used:
    - CREATE TABLE tasks (
    - id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    - name TEXT NOT NULL,
    - description TEXT,
    - dueDate DATE NOT NULL,
    - completed BOOLEAN DEFAULT FALSE,
    - user_id UUID REFERENCES profiles(id)
    -);
- 'id' is used to give each added task a random 'uuid' so that the client and server can interact. The 'id' is the primary key.
- 'name' is for the name attribute that is the name of the task and accepts text.
- 'description' is the task description and accepts text.
- 'dueDate' is the due date of the task and accepts a date in the YYYY-MM-DD format.

- 'completed' is a boolean and accepted as bool in the table, for showing whether or not the task is completed.
- 'user_id' is a uuid and references the profile id in a foreign key. This is so the tasks are stored to the user who is logged in and pulls the uuid from the profiles table.
- Editing is also available, when the user clicks a task and presses 'edit' it takes them to another screen that has text inputs and they can update their task attributes. Once the user presses update the table is updated on both the client and server side.
- Unfortunately, you have to reload the application to see the updated and added tasks after they are updated or added.
- Another thing that I will look into fixing is the Row-level-security feature, which only works when disabled.
- Now that the tasks are able to be stored in the table, we were able to pull each user's tasks and display them on the homepage in the tasks shortcut section.

User Management

Account Creation

When loading the app, if the user doesn't already have an existing account, they can choose to create a new one on the "Create Account" screen. This screen prompts users to enter their name, email twice to confirm it, and their password twice to confirm it. The validity of the email and password are checked by the functions below:

```
const validateEmail = (email) => {
  // Perform email validation here
 // For example, check if it's a valid email format
  return /\S+@\S+\.\S+/.test(email);
};

const validatePassword = (password) => {
  // Perform password validation here
  // For example, check if it meets certain criteria
  return password.length >= 6;
};
```

An error is shown to the user if the validateEmail or validatePassword functions come back as false. Another set of errors can appear for the user on this screen, and they are for if the user's email and confirm email inputs don't match, and the same goes for the password and confirm password inputs.

```
const handleBlurConfirmEmail = () => {
  if (email !== confirmEmail) {
```

```
      setConfirmEmailError('Emails do not match');
    } else {
      setConfirmEmailError('');
    }
  };
```

Upon submission of the form, the account creation process is done through Supabase's Authentication system, using the supabase.auth.signUp() function. The user's email and password are sent to Supabase's authentication API, which validates the information on the server side, to ensure an account with that information doesn't already exist and that the password meets the complexity required. For the moment, our password requirements are that it has to be at least 6 characters, but to ensure account and user safety, we will be changing it to require more complex passwords during our testing/debugging phase.

```
  const {
      data: { session },
      error,
    } = await supabase.auth.signUp({
      email: email,
      password: password,
      options: {
        data: {
          name: name
        },
      },
    })
```

If the validation is successful, the user will be notified on the client-side that their account has been successfully created, and they will be taken to our app's home page. The name the user entered while creating their account can't be stored in the Users table for authentication, so it is saved to a table called "Profiles", that saves extra metadata about the user, and is linked to the individual user's uuid.

Signing In

The signing-in process is similar to the account creation process, as it is also done through Supabase's authentication API.

```
async function handleSubmit() {
    setLoading(true)
    const { error } = await supabase.auth.signInWithPassword({
      email: email,
      password: password,
    })
    if (error){
```

```
    alert(error.message, error.status)
    console.log(error.message, error.status)
  }
  setLoading(false)
if (!error){
  goToHomePage();
}
}
```

Changing User Information

At the moment, the user can only successfully change their password. Due to unknown issues with our external SMTP server, we aren't able to send users emails to verify their account on sign-up. Because of the issues with the emails, and Supabase's restrictions on changing user information while unverified, the "Update Email" option in the Settings page is not working correctly at the moment.

However, the user is able to change the password associated with their account, and the change is reflected in the database as soon as they confirm the change in the app.

```
const handleUpdatePassword = async () => {
  try {
    if (isValidPassword) {
      console.log('New Password:', newPassword);
      await supabase.auth.updateUser({
        password: newPassword,
      });
      console.log('Password updated successfully');
      closeUpdatePasswordModal();
    }
  } catch (error) {
    console.error('Error updating password:', error.message);
  }
};
```

Testing and debugging

The testing during this phase of our app was an ongoing and crucial process during the construction of our back-end. There was a lot of trial and error and we built different portions of our backend and we conducted simultaneous testing during each phase. The initial phase we focused on was the construction of the user profile tables in our database. As we worked to create the database and table for our user profiles, users tested the application's create

account/login screen. The task given was to have multiple accounts created. From there, we refreshed our database to make sure that changes were being saved. The following challenge was to make sure that the user could login using their previously created account. During this testing, it was also crucial to make sure our form was not allowing any logins that were not in existence in the database. Our team managed to create a successful result in this area and the user is able to be sent to our home page upon login. Once on the home page, users are able to see the widgets for the guides, calendar and tasks portions of the application. The user's name is also displayed on the top of the user's home page.

While our group had much success in terms of the initial user profile setup, each portion of the application does have much debugging to undergo. Currently, the user is not able to favorite guides. For the tasks functionality of the application, the user is in fact able to successfully add tasks, mark them complete and they are both displayed to the front-end and simultaneously saved to the database. The issue with this section of the application is that our back-end does not yet update any changes on the front-end when existing tasks are edited. This is one particular issue that we will need to focus on in any further debugging as we were not able to successfully provide this function as of now. The calendar function in the application allows for the user to create and save multiple events to our database, however, the functionality for it to post to the calendar itself has not been added. This is also a feature that will need to be tackled in the next phase.

As for the user profile settings, the user is able to successfully logout, log back in, and change their password inside their profile. Currently, the user is not able to change their email and successfully log back in with the newly updated email address. Overall, there are lots that we were successfully able to create and save to our backend as a team or primarily researchers and designers. For this next phase of debugging our application, we plan to put these issues at the forefront of our agenda to be able to provide full functionality of our application to our user audience.

Issues that need to be resolved:

- Unable to update name or email associated with account
- SMTP not sending verification/confirmation emails
- Calendar page connected to database
- Favorite Guides page connected to database
- Connecting Tasks/Calendar completely to homescreen

Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

Testing and Debugging

Executive Summary

For our 'Testing and Debugging" milestone, we once again split up our tasks into the respective sections of our application. This has proven to be the most effective way to handle tasks and efficiently each section throughout each milestone. We had a set list of bugs we wanted to sort out and prioritize during this time period. There was at least one thing that needed to be fixed in each section of our application.

Tasks needed to fix the asynchronous update as the user would add, delete or update a task. Calendar needed to achieve getting the added event to display on the front end and not only the backend. The Profile page has prioritized confirming email and changing their email. The Home page was updated to make the tasks display on the front end of the homepage and have the calendar update. Finally, the Guides are being worked on to make sure the user is able to have a suggested or favorite guide.

As a team we have put in a valiant effort to make sure we can get as much done as possible and we have achieved all we can within time and skill restraints. Our application is now even closer to being fully functional and can handle the basic tasks we originally wanted users to be able to complete. Of the problems listed above all have been resolved and will be tweaked finally in the upcoming week before deployment. The outstanding issues are miniscule and again, everything we have prioritized is within our current scope of the project. We have sacrificed some of our design to make sure functionality came first.

As far as testing, we decided to use a small group of individuals to complete a select group of tasks. This was documented in a video followed by an analysis using the 'Think Aloud Protocol." Testing proved to be an important aspect of seeing where we needed to fix certain functionality as the users were seeing the completed application for the first time.

Since we are nearing the end it has taken a lot of determination to stay focused and make sure we produce the best work we can. Having our database working has been one of our greatest achievements as it is the backbone for having unique and personal accounts since everything is linked to specific user ids. While that was a major hurdle, we streamlined everything after it was resolved.

Group 1: Sara Schwartz, Lisa Heinzman, Shakeera Singh, Ja'Mecia Rosier, Jacob Rothstein

Final Walkthrough of "HiveHelp"

(link placeholder will upload in a few)

Test Use Cases

1. **Create new account**
    a. Open application
    b. Select "Don't have an account" Click here
    c. Fill out information fields
    d. Verify email
    e. Return to login
2. **Change Color Palette to Dark Mode 1**
    a. Open application
    b. Select User Profile icon from Nav Bar
    c. Select Settings
    d. Set Color Mode
    e. Select Dark Mode 1
    f. Confirm set color palette
3. **Change Color Palette to Light Mode 2**
    a. Open application
    b. Select User Profile icon from Nav Bar
    c. Select Settings
    d. Set Color Mode
    e. Select Light Mode 2
    f. Confirm set color palette
4. **Select the Third Self Care Guide in the Menu**
    a. Open application
    b. Select Guides icon from Nav Bar
    c. Select the Self Care section from Guides main menu
    d. Scroll (top → bottom) to the third guide in the listings
    e. Select the third guide to open

5. **Favorite the fourth guide in the Work guides section**
    a. Open application
    b. Select Guides icon from Nav Bar
    c. Select Work Guides category
    d. Scroll to the fourth guide in the category
    e. Select the "Favorite" icon
    f. Return to app home screen to see the Favorite guide appear on the Guide widget
6. **Change email address**
    a. Open application
    b. Select User Profile icon from Nav Bar
    c. Select Settings
    d. Select Login info
    e. Go to email address
    f. Change email address
    g. Save changes
7. **Edit the note in the third task in the user's list**
    a. Open application
    b. Select Tasks from home screen widget
    c. Scroll to third task in user's list
    d. Select task
    e. Select edit note
    f. Change note to (insert note)
    g. Save change
8. **Change the user's password**
    a. Open application
    b. Select User Profile from Nav Bar
    c. Select Settings
    d. Select login info
    e. Set user name
    f. Save Change

9. **Create a calendar event**
    a. Open application
    b. Select calendar widget from Home Screen
    c. Select calendar event on May 15 2024
    d. Select delete
    e. Return to calendar
10. **Check off the fifth task**
    a. Open application
    b. Select task widget from home screen
    c. Scroll to fifth task
    d. Select the empty checkbox on the fifth task
11. **Edit an Existing Task's Description**
    a. Open the application
    b. Select the Tasks section from the nav bar
    c. Select a previously created task "Cook Dinner - Chicken Caesar & Lasagna"
    d. Edit the task's description to include additional details from recipe
    e. Save changes
12. **View completed Tasks**
    a. Open application
    b. Select Tasks widget from home screen
    c. Scroll down to bottom of Tasks Page
    d. Select view completed Tasks
13. **Logout and Create New Account**
    a. Open application
    b. Select User Profile
    c. Scroll down User Profile Page
    d. Select Logout
    e. Select "Click here" under "Don't have an account? "
    f. Enter in a username, password, and email
    g. Select create account
14. **Change Calendar Month to March**

a. Open application
b. Select calendar from nav bar
c. Click the right arrow till month swaps to May

15. Add a New Task

a. Open application
b. Select tasks from nav bar

16. Delete the Most Recent Task

a. Open application
b. Select tasks from homepage
c. Select the task you want to delete
d. Select delete task

Testing Approach

The testing of our app is being conducted utilizing the think-aloud protocol. Each of us will give our users a set of tasks out of our pool of cases to perform while using the app. We will then record their screens and faces/voiceovers via Zoom and IPhone Screen Record. As they go through it we will take notes as needed. After conducting the video, we will perform a task analysis where we will record the number of errors and points of confusion from the user think-alouds. We will also be recording any feedback about the overall design of the app and what functions could be added, taken away, or changed. These results will be compiled into a final tally and recorded in the defect report. Following the testing, we will be conducting debugging for our application in response to each of the users' feedback.

User Profiles

1. **Zameena Singh**

Age: 24

Education: Currently

Occupation: Store Manager at a Cigar Lounge and Cafe

Location: Kissimmee, FL

Background and Goals: Zameena is a 24 year old woman working full-time after the completion of her Associate's degree. She is contemplating changing jobs and going back to school for her Bachelor's degree. This app would be useful as she manages her apartment, relationships, and future job/educational endeavors.

Link to User Test: https://youtu.be/aPQlbmGZXak?si=F6Ro1n3Hfm-pfxFX

2. **Connor Weichman**

Age: 24

Education: AA

Occupation: Unemployed

Background and Goals: Connor is going back to school to receive his Bachelor's degree.

Link to User Test: https://youtu.be/4-QDyD4Z9Zw

3. **Daniel Vanderbrink**

Age: 26

Education: AA

Occupation: Full-Time Customer Service Representative

Background and Goals: Daniel is currently working full-time and studying part-time to become a web developer. He would like to have a way to organize his tasks and important events to fit his work and study schedule.

Link to User Test: https://youtu.be/lq2tevA6mOE

Task Analysis Reports

1. Zameena: (error count = 3)

1.1. Sign in: (00:00-00:08)

1.1.1. Logged in to app with existing account

1.2. Find 3rd self care guide: (00:18-00:50)

1.2.1. Found guides page immediately

1.2.2. User took time to figure out where to find self care guides

1.2.3. Inferred it would be under "Personal"

1.2.4. Opened guide category page

1.2.5. User found "Skin Care", the third page in the section

1.3. Change user password: (00:55-01:14)

1.3.1. Found profile page immediately

1.3.2. User navigated to settings page promptly

1.3.3. Entered new password

1.3.4. Selected update to save new password

1.4. Mark 5th task complete: (01:20-01:30)

1.4.1. Found tasks immediately

1.4.2. Noted it was found by identifying notepad icon

1.4.3. User found 5th task "car wash"

1.4.4. Selected hexagon to mark completed

1.5. View completed tasks: (01:45-01:48)

1.5.1. User was already on tasks page

1.5.2. Navigated using "Show Completed Tasks" button

1.5.3. User noted that the "car wash" task appears in the completed section

1.6. Move to March on the calendar: (01:55-02:02)

1.6.1. Identified and selected calendar icon in nav bar

1.6.2. Selected left arrow to move back a month on the calendar

1.7. Add a new task: (02:09-02:43)

1.7.1. Selected tasks from the nav bar

1.7.2. Selected the "+" button in the top right corner

1.7.3. Filled out the task title, description and due date

1.7.4. User did not format the due date properly

1.7.5. Selected add task

1.8. Change the color palette: (02:50-03:09)

1.8.1. Selected the profile page

1.8.2. Selected the "change color theme" button

1.8.3. User selected Dark Mode 2 initially

1.8.4. User did not like that mode

1.8.5. Selected Dark Mode 1 instead

1.9. Logout: (03:11-03:16)

1.9.1. Selected the profile icon again

1.9.2. Pressed the logout button

2. Connor: (error count: 3)

2.1. Create a new account (00:09-01:38)

2.1.1. User filled out sign in instead of selecting "Don't have an account? Click here" at the bottom
2.1.2. User redirected to "Don't have an account? Click here"
2.1.3. User filled out all information fields
2.1.4. User selected sign in
2.1.5. Verify email popped up on the screen
2.1.6. User navigated to their email to confirm
2.1.7. User returned to app to sign in
2.1.8. User was able to sign in to the app

2.2. Change color theme (01:40-02:27)

2.2.1. User navigated to the calendar button on the nav bar
2.2.2. Navigated to the guides page through the nav bar
2.2.3. User selected the profile page
2.2.4. Selected the "change color theme" button
2.2.5. User selected Dark Mode 1

3. Daniel (error count: 1)

3.1. Sign in (00:08-00:24)

3.1.1. Logged in to app with existing account

3.2. Find 3rd self care guide: (00:25-00:44)

3.2.1. User navigated to the tasks page
3.2.2. Navigated to the guides page through nav bar
3.2.3. User selected personal guides immediately
3.2.4. Successfully selected the 3rd guide in this section
3.2.5. Browsed the content reading it

Debugging

Email Verification

The issue of emails not being sent to the user has been resolved by changing the SMTP provider from Resend to SendGrid. For some reason, the initial SMTP provider just wasn't working with our server correctly to send users confirmation emails on signing up and when they change their email associated with their account.Since making the change to the new provider, notification emails are being sent properly.

Tasks

During the debugging, there was an issue that users had when adding a task and having to refresh the page. We have completely fixed the issue of the users adding a task and having it show up immediately. There is still a refresh issue when the user is to update or delete a task, however, it is functional when it comes to updating or deleting from the database. This was an issue we deemed fixed and not a deal breaker within the scope of our application as the user can add and see the new task immediately. Taking the extra step to logout and log back in to see the update or delete seemed doable as users would likely log out and back in as they were to close the app and reopen it anyway. Corrected the way the show and hide completed tasks function works when users mark the task completed and uncompleted. There was an issue with the tasks not being marked complete and uncomplete, the tasks would bump other tasks back and forth rather than just moving the singular task. We were able to correct this and it now works from task to task. The main functionality of tasks for users is fully working and is even displayed in a scroll list on the homepage displaying the title of the task like a notepad.

Guides

Regarding the guides page, the two main issues we had were the display of a suggested guide when the button on the user's homepage was selected and the save and display of a user's favorites guides. In repairing these bugs, the guides were added into our database so we could properly save the user's favorite guides. Currently the suggested guide is still under debugging as we were not able to fully resolve this issue. As of now it suggests a guide category, and not one single guide in particular.

Calendar

The primary issue concerning the Calendar portion of the "Hive Help" application was the display of calendar events. At the beginning of this phase, the application was only saving the user's added events on the backend to our supabase table. However, it was not saving the added event to the user's account properly and was returning the value of uuid as null. In repairing this, the current events on the user's calendar had to be properly called from the json data rather than having been hard coded. Following that, we worked to get the Calendar to read and display the database data that each user adds to their calendar. Currently, this main functionality is resolved. While the user does need to select the back button to view the added event, it is working as needed.