



GIT HYPE

GIT HYPE TEAM



Sanjay
Mylanathan

Giselle
Wang


Denise
Ou

Pantysh
Ghurburrun

Yuxin
Chen

Alex
Russell

Jawad
Arshad

The background is a dark blue, almost black, space filled with intricate, glowing blue geometric patterns. These patterns consist of various lines, rectangles, and trapezoids that create a sense of depth and perspective, reminiscent of a futuristic architectural interior or a complex digital circuit. The lines are sharp and have a slight glow, giving the impression of light reflecting off metallic surfaces. In the center of the image, the word "HYPE" is written in a bold, white, sans-serif font. The letters are thick and blocky, with a slight shadow that makes them stand out against the dark, complex background. The overall composition is balanced and visually striking, with the central text acting as the focal point amidst the intricate geometric design.

HYPE

Problem



Students

- Want hands-on experience & to explore different career options
- Want to collaborate with like-minded individuals & expand their network

Startups

- Want innovative, driven, capable & motivated individuals
- Has a tight budget & small team

LinkedIn & Co-ops

- Unable to help form personalized connection
- Opportunities restricted to certain qualifications, cities, or regions



Hype

Web Application

- Connects students with startups



Opportunities

- Competitions & internships

Interactive Platform

- Communicate, collaborate & innovative



Target Users



- Undergraduate (including recent graduates)
- Innovative & highly motivated
- Team player
- Desire to learn & curiosity in exploring different career options

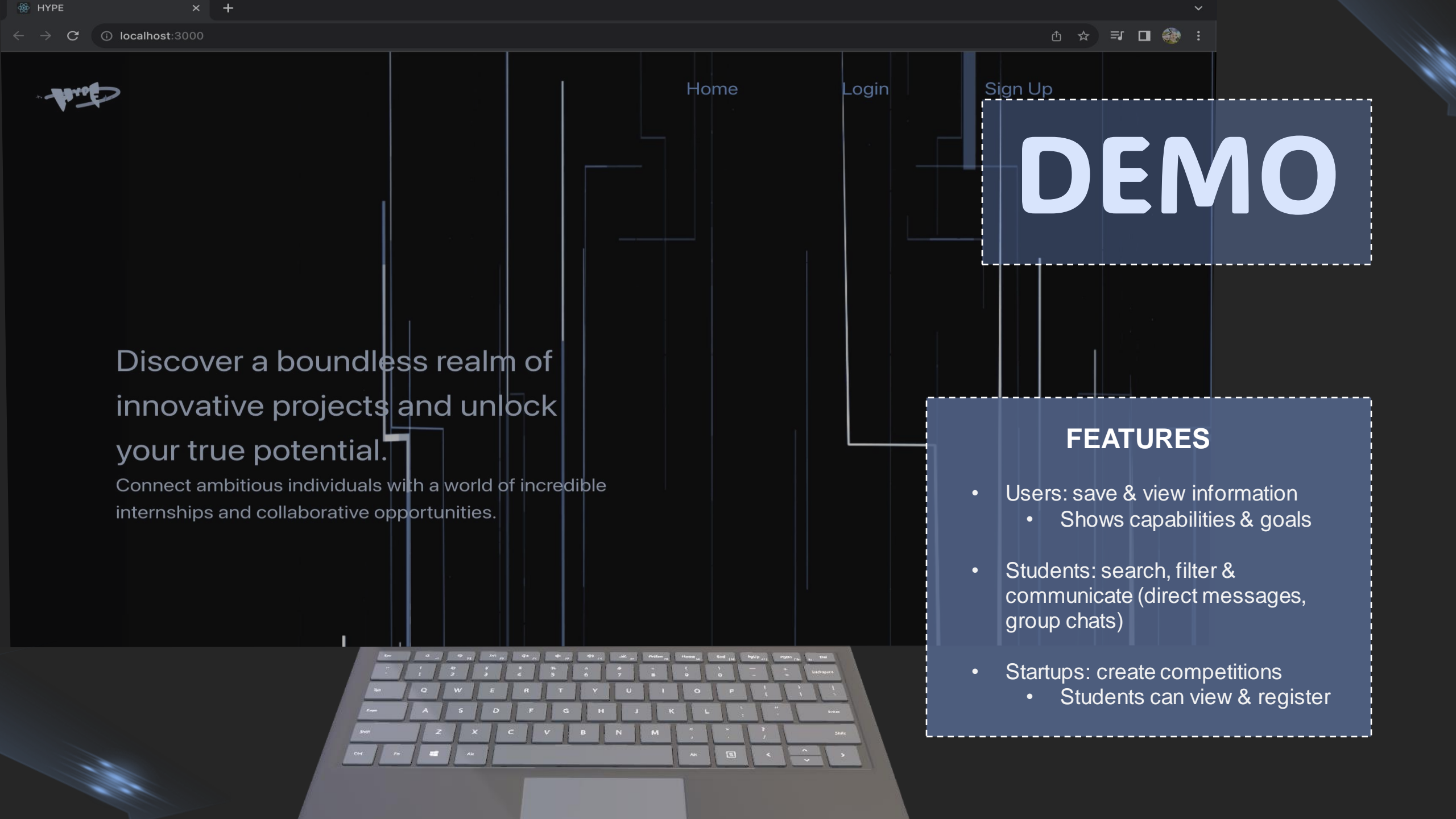
Students



- Good team dynamics (collaboration focused)
- Flexible structure
- Passion for innovation & technological advancement
- Has experienced individuals in multiple different fields

Startups



[Home](#)[Login](#)[Sign Up](#)

DEMO

Discover a boundless realm of
innovative projects and unlock
your true potential.

Connect ambitious individuals with a world of incredible
internships and collaborative opportunities.

FEATURES

- Users: save & view information
 - Shows capabilities & goals
- Students: search, filter & communicate (direct messages, group chats)
- Startups: create competitions
 - Students can view & register



PROCESS DISCUSSION

TEAM WORKFLOW

Communication

In-person Meetings



Discord



Whatsapp



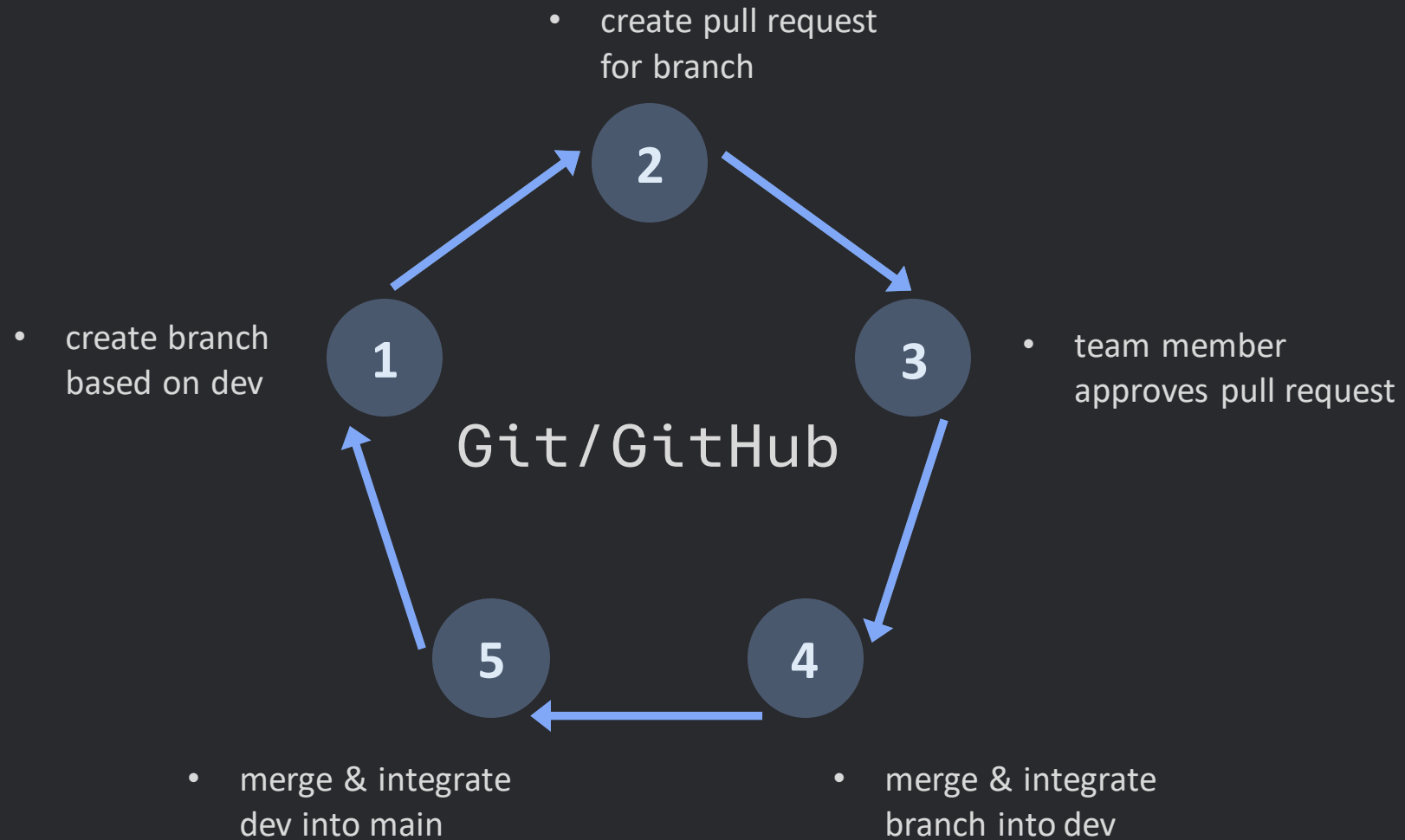
Zoom



Slack



TEAM WORKFLOW



TEAM WORKFLOW

Division of Tasks

Sprint 1

- Created Jira tasks
 - Did not assign story points
 - Selected 11 tasks based on priority
- Individuals assigned Jira tasks to themselves
 - Minimum 1 task each
- Unassigned Jira tasks picked up later in sprint

Sprints 2-4

- Assigned story points to Jira tasks
 - Around 40 story points per sprint
 - Selected tasks based on priority
- Individuals assigned Jira tasks to themselves
 - Average 6 story points
- Completed unfinished Jira tasks from previous sprint

DECISIONS

Good Decisions

- Creating Figma prototypes of our vision before implementation
 - Saved time
 - Ensured consistent design layout
- Redefining Jira tasks to be smaller & more fine-grained
 - Allowed better division of work
 - Ensured manageability & completion of tasks
- Changing the definition of done for Jira tasks to include merging & integrating code with dev branch
 - Minimized conflicts
 - Ensured features can be released on time

Bad Decisions

- Created Jira tasks that were too big
 - Difficult to manage
 - Increased individual workload
 - Impacted quality of work
 - Resulted in incomplete tasks



TECHNICAL DISCUSSION

Adding chatId in create chat API POST Response

```
1 import { POST, chatEndpoint } from "../endpoints"
2 import { fetchCall } from "../fetchCalls"
3
4
5 export const createChatSaga = async (chatname, othersUserId, authToken, onSuccess, onFailure) => {
6   const response = await fetchCall(chatEndpoint, POST, { chatname, othersUserId, authToken });
7
8   const data = await response.json();
9
10  if (data.success) {
11    onSuccess(data);
12  } else {
13    onFailure(data);
14  }
15 }
```

Issue: the absence of chatId in the response of the Create Chat POST request.

- Only a message indicating the success of the chat creation process.
- Hindered the smooth integration of the chat creation process with other functionalities.
- chatId is a mandatory body in update chat message.

```
84
85 router.post('/', async (req, res) => {
86   try {
87     const userId = req.userId;
88     const { chatname, othersUserId } = req.body;
89     const { success, message } = await createChat(chatname, userId, othersUserId);
90
91
92     if (success) {
93       res.status(200).json({
94         success: true,
95         message: message,
96       });
97     } else {
98       res.status(400).json({
99         success: false,
100        message: message,
101      });
102    }
103
104    } catch (err) {
105      console.log(err);
106      res.status(500).json({
107        success: false,
108        message: "Internal server error",
109      });
110    }
111  });
112 }
```

Adding chatId in create chat API POST Response

```
6 const response = await fetch(endpoint, req); // { chatname, othersUserId, chatId };
7
8 const data = await response.json();
9
10 if (data.success) {
11   createChatUpdate(data.chatId);
12   onSuccess(data);
13 } else {
14   onFailure(data);
15 }
16 }
```

Solution:

- Backend: chatId was added to the response of the POST request.
- Frontend:
 - Use createChatUpdate() to save the chatId from the API response in the saga file.
 - Add the function useCreateChatContext() to get the chatId in the popup window frontend javascript file, so that when the response is returned, we could use this chatId in the update chat message request.

```
router.post('/', async (req, res) => {
  try {
    const userId = req.userId;
    const { chatname, othersUserId } = req.body;
    const { success, message, chatId } = await createChat(chatname, userId, othersUserId);

    if (success) {
      res.status(200).json({
        success: true,
        message: message,
        chatId: chatId,
      });
    } else {
      res.status(400).json({
        success: false,
        message: message,
      });
    }
  } catch (err) {
    console.log(err);
    res.status(500).json({
      success: false,
      message: "Internal server error",
    });
  }
});
```

Store Type Of User Account Locally

```
if (localStorage.getItem("NavType")) {  
  type = JSON.parse(localStorage.getItem("NavType")).storeType;  
}
```

- **Issue:** Navigation bar required an account type after login
- **Attempted solution:** Assign types as props to Nav component
- **Challenge:** Shared web pages between student and startup users; prop addition insufficient
- **Alternative solution:** Use localStorage to share type data across pages
- **Rationale:** User type data is non-sensitive and simple
- **Implementation:** Added code after successful user login
- **Result:** User type stored in localStorage for later use in Nav component

```
const loggedIn = useAuthTokenContext();  
useEffect(() => {  
  if (loggedIn) {  
    if (props.isStudent) {  
      const NavInfo = { storeType: "student"};  
      localStorage.setItem("NavType", JSON.stringify(NavInfo));  
      history.push("/student-profile"); //change to where you want to go  
    } else {  
      const NavInfo = { storeType: "startup"};  
      localStorage.setItem("NavType", JSON.stringify(NavInfo));  
      history.push("/startup-profile"); //change to where you want to go  
    }  
  }  
}
```



The background is a dark blue, almost black, space filled with glowing light blue lines and geometric shapes. On the left, there's a vertical structure with a glowing blue light source. On the right, there's a more complex, angular structure with multiple glowing blue lines and a small cluster of bright blue dots at the top. The overall aesthetic is high-tech and digital.

SOFTWARE ARCHITECTURE

Frontend

Main Components

Components Folder

- houses individual UI components

Context Folder

- manages global state & shared data

Pages Folder

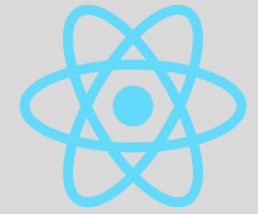
- represents different views/screens users interact with

Sagas Folder

- centralizes API calls (ensure consistency & reusability)



Frontend



Interactions

- Each page – constructed using multiple components from different folders
- Pages – access global data via contexts (e.g. user authentication tokens)
- Data fetching – pages & components utilize sagas to interact with backend

Technologies

- **styled-components** – used to style UI
- **axios** – used to make API calls
- **bootstrap** & **mui** – integrated for rapid UI development

Backend

Main Components

Models Folder

- defines data structure & interacts with database

Routes Folder

- specifies available API endpoints

Services Folder

- contains core logic for each endpoint

Utils Folder

- offers utility functions (code reusability & efficiency)



Backend



Interactions

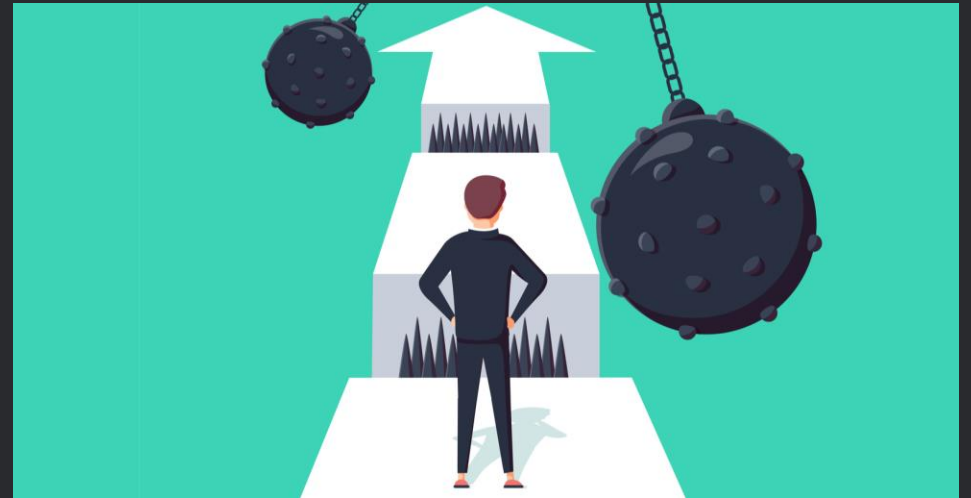
- Incoming requests – directed by routes to appropriate service
- Services when needing data operations – interact with models
- Utility functions (from Utils folder) – used to help with various tasks

Technologies

- Backend runs on **Express.js** (**Node.js** framework)
- **Mongoose & MongoDB** – database interactions
- **Nodemon** – auto-restarting server on code changes during development

Challenges/Techniques

- Achieving a responsive design
 - Especially for mobile views
- Managing global state using contexts
 - Required careful planning to avoid over-complication
- Large queries with useless data
 - As user base expanded, redefined database queries ensuring only necessary data was fetched
- Authentication & request parsing tasks
 - Implemented middleware in Express



Individual Contributions

Alex

- Setup sagas
- Helped early merge conflicts
- Profile edition
- Fixed formatting and display bugs

Pan

- User logins
- Hashing passwords
- Profile icons
- Student List with filtering

Katy

- Home Page
- Navigation Bar
- Message display
- Chat delete

Giselle

- Startup user login
- Student user signup
- Message Popup window
- Delete messages
- Competition signup
- Registered student list for competition

Jawad

- Chats feature backend
- Frontend for the login/signup form
- Chat create/edit group
- Competition page

Denise

- Design/layout: student profile & student list
- Cue card student profiles
- Customized scrollbar
- Search ability in chats
- Display icons: direct message vs group chat

Sanjay

- User logout
- Student/startup edit profile
- Creating competitions
- Design improvements & error fixes

The background is a dark, almost black, space filled with a complex network of thin, light blue lines. These lines are mostly vertical, creating a sense of height and structure. Interspersed among these lines are several bright, glowing blue streaks that appear to be light trails or energy pulses, adding a dynamic and futuristic feel to the composition. The overall aesthetic is clean, modern, and high-tech.

Thank you!