TigerConnect

Product Guide

Tajreen Ahmed • Jessica Edouard • Kevin Finch • Lillian Meng

Special Thanks to
Professor Brian Kernighan
Jérémie Lumbroso (TA Advisor)

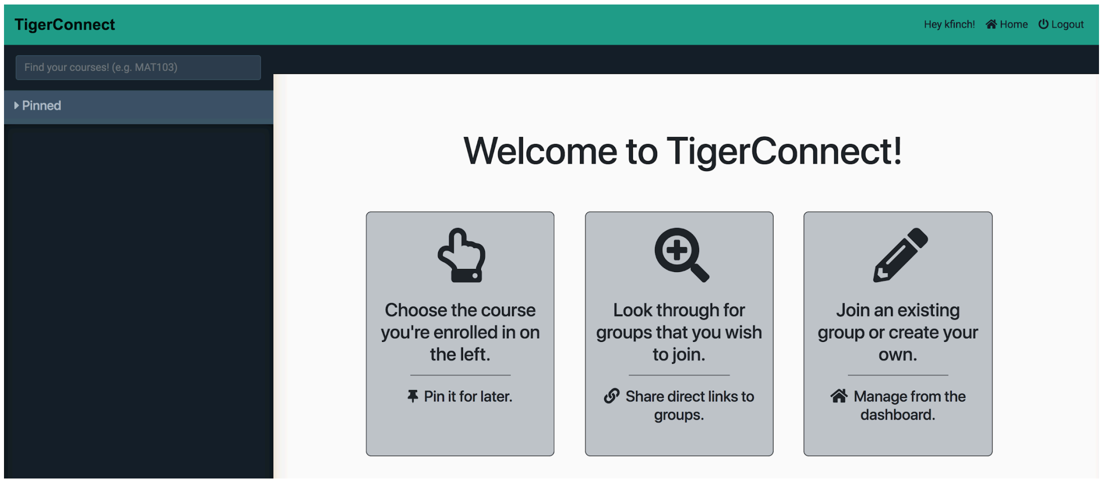
User Guide

Getting Started

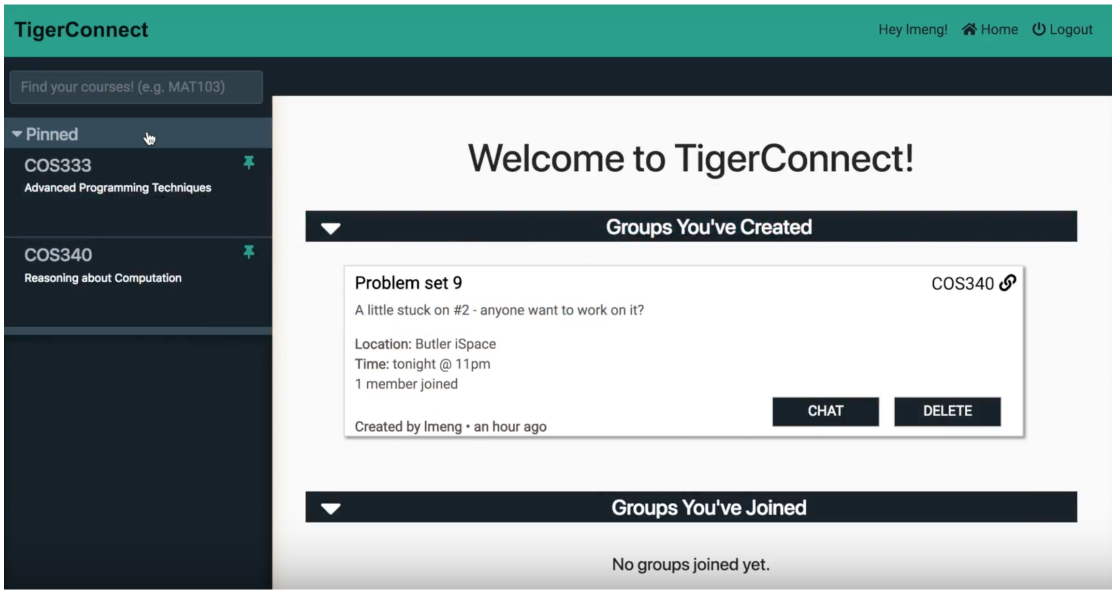
TigerConnect is a web application which can be reached at tigerconnect.herokuapp.com. The site requires the user to login through Princeton University's CAS system. Click on the login button to login and then proceed to the app.

Viewing Your Dashboard

Once you login, you will see the dashboard page, with some onboarding instructions giving a quick overview of TigerConnect features.


TigerConnect
Hey kfinch!
Home
Logout
Find your courses! (e.g. MAT103)
Pinned
Welcome to TigerConnect!
Choose the course you're enrolled in on the left.
Pin it for later.
Look through for groups that you wish to join.
Share direct links to groups.
Join an existing group or create your own.
Manage from the dashboard.

If you're a returning user who has already created and/or joined groups, your dashboard will give you easy access to those groups instead.



Problem set 9 COS340

A little stuck on #2 - anyone want to work on it?

Location: Butler iSpace
Time: tonight @ 11pm
1 member joined

CHAT DELETE

Created by lmeng • an hour ago

No groups joined yet.

Finding Your Courses

Begin by searching for courses that you want to find problem set (p-set) groups for. The bar on the left will display search results for courses in the current semester. You may find it helpful to pin the courses you're enrolled in for easy access later. Select a course in the left-hand bar to see any p-set groups that may already exist for that course.

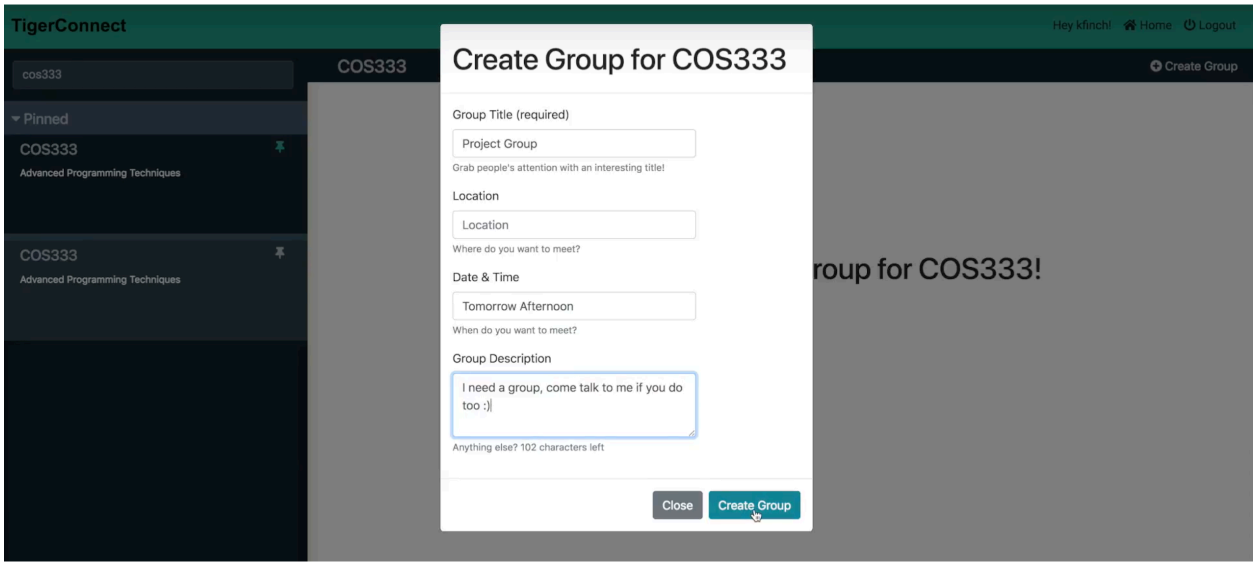
Problem Set Groups

Joining Existing Groups

Once you've selected a course on the left-hand bar, any existing groups created for that course will show up in the main panel. Browse through and **Join** any groups that look good to you. Once you've joined the group, you can enter the group chat (more on this in a moment).

Creating New Groups

If you don't see a group that fits your needs, create your own! The **Create Group** button at the top right opens a form. Give your group an informative title and description to help others find your group. The location, time, and description fields are optional, so if you're not quite sure yet, feel free to leave them blank. Once your group has been created, you'll be able to chat with any other students who join your group and figure out more detailed meeting logistics from there.



Create Group for COS333

Group Title (required)
Project Group
Grab people's attention with an interesting title!

Location
Location
Where do you want to meet?

Date & Time
Tomorrow Afternoon
When do you want to meet?

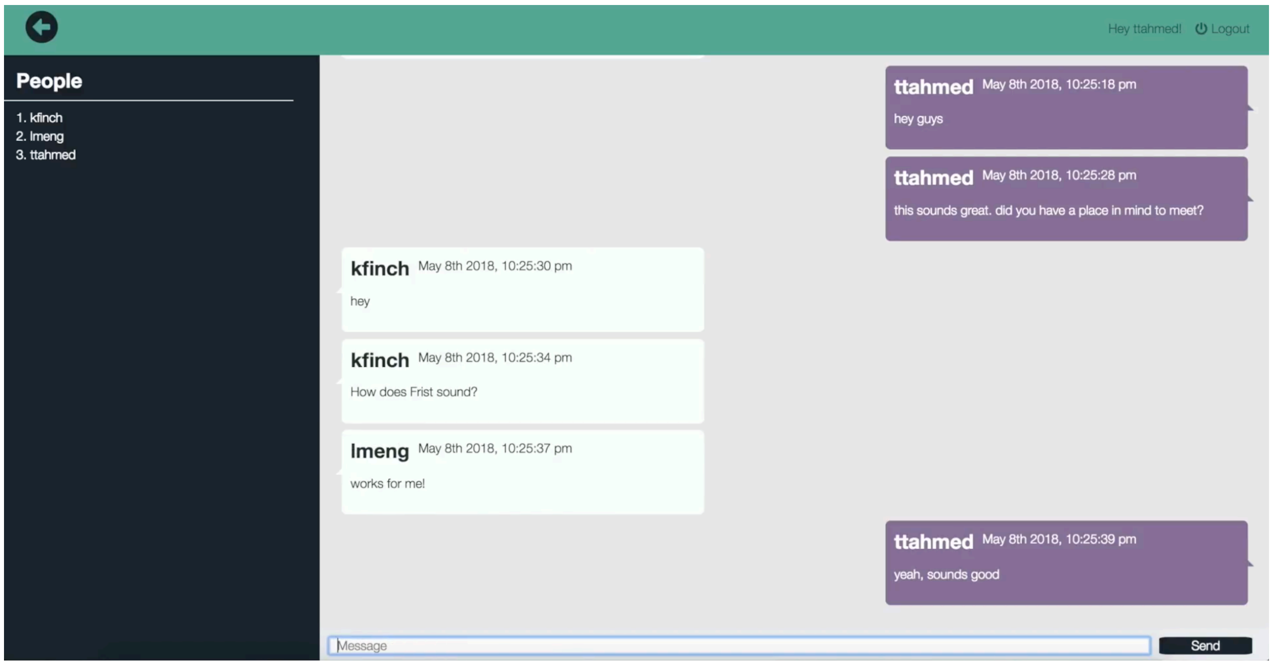
Group Description
I need a group, come talk to me if you do too :)
Anything else? 102 characters left

Shareable links

Once you've found a great group (or just created your own), share it! Each group has a unique shareable link. Click on the icon to get a copyable link to share. When someone logs in and visits the link, they will be directed immediately to that specific group, where they can join and see the chat.

Live Group Chat

Once you're a member of a group, you'll be able to click **Chat** to see a live group chat with all the members in that same group. There may be students in the group that you haven't met before — just click their netid in the left bar under **People** to see their TigerBook profile. Use the group chat to flexibly figure out group meeting details that work best for everyone.



ttahmed May 8th 2018, 10:25:18 pm
hey guys

ttahmed May 8th 2018, 10:25:28 pm
this sounds great. did you have a place in mind to meet?

kfinch May 8th 2018, 10:25:30 pm
hey

kfinch May 8th 2018, 10:25:34 pm
How does Frist sound?

lmeng May 8th 2018, 10:25:37 pm
works for me!

ttahmed May 8th 2018, 10:25:39 pm
yeah, sounds good

Leaving & Deleting Groups

Hopefully your group met and you guys finished that p-set — nice! If you'd like to keep meeting with that group, all your information is still saved in the group, so just navigate back to your dashboard to see previous chat messages and group details.

If this was a one-time group, feel free to **Leave** the group. This will remove it from your dashboard. If you are the creator of a group, instead of leaving, you are able to **Delete** that group entirely. Be mindful of deleting groups that other group members may still want. Otherwise, leave the group as-is, and it will remain until all groups are cleared at the end of each semester.

•••

You're all good to go with TigerConnect. Good luck with those p-sets!

Developer Guide

Getting Setup

In order to set the project up for local development and testing, clone the GitHub repository at https://github.com/KevinMFinch/TigerConnect. When you clone the repository, run `npm install` and `npm start`. You will now have a local version of the project running, which can be accessed at localhost:5000.

Backend Stack (./server/db/mongoose.js, ./server/index.js)

The backend uses a node.js server and a MongoDB database. On top of these base technologies we use Express.js and Mongoose. Express is a popular routing library for node and Mongoose is an object-relational mapper which enforces rigid documents and provides validation upon insertion into the database and querying. For the real time chat portion of the application, we use an npm library socket.io which provides a simple, event driven socket programming library. This allows real time, two way communication between the server and the client. We also use a powerful utility library called Lodash to combine database query results without duplication.

Models (./server/models/)

Using Mongoose, we defined four models which exist in the Mongo database as collections. The four models we defined are

1. **Course**. This model contains relevant information for the courses. The course information is scraped from Princeton's webfeed for course offerings for the current term. Contains course title, department code, course number, and any cross listings.
2. **CourseEvent**. The name is a relic from when we originally planned to expand to have groups for other things like recreation. Contains title, description, location, time, and creator of the group, the id of the course that the group is for, as well as the current number of members in the group and their netids.
3. **User**. Contains netid, a list of their pinned courses, and whether or not the pinned courses section should auto-expand when their dashboard loads (marked as being kept open if the user opens the section, and marked as kept close if the user closes it). This helps us provide convenience for the user.
4. **ChatRoom**. This model contains an array of message objects (comprised of the message text, netid of the sender and the time it was sent) as well as the ID of the room. The room ID is the Mongo defined _id of the group that the chat room is for.

Scraper (./server/scraper/scraper.js)

We created our own custom scraper in node for scraping the Princeton web feed for course offerings. It pulls the relevant information from the web feed and populates our Course collection. The scraper relies on the `fs` and `http` standard libraries in node.

Routing (./server/index.js, ./server/routes)

We use Express to handle routing. This allows us to both serve pages (like /main and /chat) and provide a RESTful API to the front end of the app. The API can be accessed through several endpoints. We have several “base” API paths, and each of those base paths have their own endpoints for doing certain things relating to the base path. The base paths are /api/courses/, /api/courseEvents/, /api/auth/, /api/users/, and /api/chatRoom/. To see what endpoints are available for each base path as well as what data needs to be passed to each, check the corresponding file in the /server/routes/ directory of the project.

Front End Systems (./server/public/)

jQuery

jQuery provides a convenient and easy way to manipulate the DOM. Since we make large use of the RESTful API, we needed some way to dynamically manipulate the DOM based on the queries. We use jQuery and vanilla JavaScript to do this.

Bootstrap

Bootstrap is a super simple CSS library. It provides a grid based system, breaking the screen into 12 columns. This makes organizing page content extremely easy and allowed us to focus more time on developing the app and polishing it rather than fiddling with styles for page layout for a long time.

Moment.js

One important feature we wanted to add was to have messages that would display how long ago a group was created. The JavaScript date object is pretty clunky and does not have a very powerful set of features. In order to address this issue, we used a library called moment which simplifies working with dates immensely. It allows easy formatting of dates as well as a “time from now” feature, which is exactly what we were looking for.

Client/Server Communication (/public/chat.js, ./server/index.js)

Fetch

We made heavy use of the Fetch API, which is a built in JavaScript feature set which allows for fetching resources across the network. We used this to make our HTTP GET, POST, and DELETE calls which were received by the database through our RESTful API.

Socket.io

As mentioned above, we needed a way to have two way communication between the client and the server. We used socket.io, a socket programming library for Node, in order to do this. Socket has an npm module which is installed on the server, as well as a client side library. The client side library is then used to setup a connection to the server and they are controlled through emitting and receiving events.