NUSMentors Milestones

The application: Get started at nusmentors.com!

Milestone 1: *Describe your application and explain how you intend to exploit the characteristics of mobile cloud computing to achieve your application's objectives, i.e. why does it make most sense to implement your application as a mobile cloud application?*

NUSMentors is a real time mentorship management application that is targeted at NUS students, alumni and professors. It connects students who are looking for mentorship with people who are willing to provide mentorships. It includes a clean interface for students to request for mentorships and for mentors to accept requests.

NUSMentors takes advantage of the strengths of mobile cloud applications in multiple ways:

1. Easy to start: No one likes having to jump through the hoop of installing a native application from App Store before being able to try it. Since NUSMentors is a mobile web application, users can get started immediately! Furthermore, students do not look for mentors every day, so it is great that NUSMentors does not take up space on their phones like a native application would.
2. Productivity: Students can submit and track their mentorship requests from anywhere. Likewise, mentors can browse mentorship requests and accept them from anywhere. The user can get what they want regardless of device or platform. By removing these restrictions, NUSMentors allows the user to find a mentorship as quickly as possible.
3. Availability: After receiving the contact details of the mentee via the application, a mentor can view the contact details on the move. Even when the internet connection is poor or non-existent, the mentor can still view the contact details because NUSMentors is a mobile cloud application. The mentee can do the same using the contact details of the mentor!

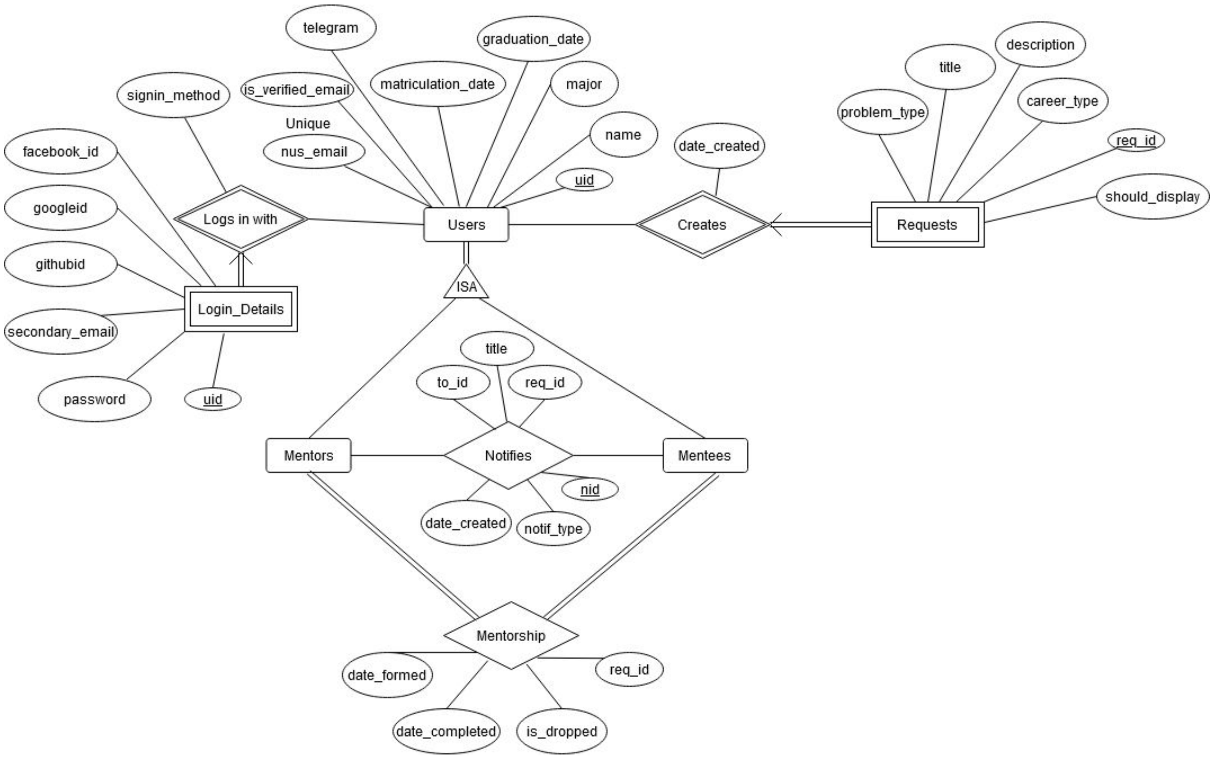
Milestone 2: *Describe your target users. Explain how you plan to promote your application to attract your target users.*

Our target users are NUS students, alumni and professors.

NUS students, especially NUS Computing students, are the target “mentees”. They are interested in getting career advice for careers in private sector or academia, they may even appreciate some help with interview preparations. We can attract these NUS students through events like Computing Day, NUS Hacker events and orientation camps.

NUS seniors, alumni and professors are the target “mentors”. Their motivations may different. For seniors, they may be influenced by a pay-it-forward ecosystem. NUS Alumni may be interested to network with juniors in an authentic way. This allows them to attract young and plucky talents for their start-ups or companies. NUS Professors may be interested to pass down their knowledge or hire students to help with their research. We can attract these seniors, alumni and professors by reaching out to them through Linkedin, email, or recruiting events.

Milestone 3: *Draw an Entity-Relationship diagram for your database schema.*



Milestone 4: *Explore one alternative to REST API (may or may not be from the list above). Give a comparison of the chosen alternative against REST (pros and cons, context of use, etc). Between REST and your chosen alternative, identify which might be more appropriate for the application you are building for this project. Explain your choice.*

We explored using a GraphQL API instead of REST API, although we ultimately decided to go with REST API.

REST APIs are more prone to over-fetching and under-fetching. Since REST APIs have rigid endpoints designed to return the stipulated data whenever it is queried, we may get unnecessary data (over-fetching) or make multiple calls before getting the relevant data because each individual query does not return sufficient data (under-fetching). GraphQL is more flexible and allows us to retrieve the data we need in a single API request. After defining the structure of the data we need, the server will return the corresponding data. In this way GraphQL is able to avoid over-fetching and under-fetching. This reduces the number of round-trips needed, which improves the performance.

However, when caching is used to reduce the number of API calls, REST APIs can perform better. REST APIs take advantage of the HTTP caching mechanism and returns cached response faster. GraphQL also has options for caching, but they are not as good as that for REST APIs currently. In terms of performance, it is not clear which choice is better.

In terms of security, REST APIs can be secured by implementing different API authentication methods (e.g. via HTTP authentication, JSON Web Tokens, or OAuth 2.0 mechanisms). While GraphQL also provides some measures to ensure API's security, they are not as mature as that of REST APIs.

Since NUSMentors collects contact information about participants, we felt that security was an important aspect. This is why we chose REST API over GraphQL API.

Milestone 5 (REST API): *Design and document all your REST API. If you already use Apiary to collaborate within your team, you can simply submit an Apiary link. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with an explanation on the purpose of each request for reference. Also, explain how your API conforms to the REST principles and why you have chosen to ignore certain practices (if any). You will be penalized if your design violates principles with no good reasons.*

Link to our documentation: https://documenter.getpostman.com/view/8359945/TVKFzFvo

Our API conforms to the REST Principles as detailed below.

1. Client-server architecture – In our code, the client (found in ./web) and server (found in ./api) are separated and communicate only via the endpoints exposed by the server.
2. Statelessness – The server does not store any client context between requests. Each request is self-contained and the session state is held in the client.
3. Cacheability – Responses are defined as cacheable whenever they are cacheable. Appropriate request types are used to implicitly define the cacheability of the response.
4. Layered Architecture – We use a layered architecture: API Endpoint layer, Heroku Layer, Database Layer. Additional layers can be added as needed without affecting communications or needing update in client code or server code.
5. Uniform Interface – Resources are identified through URIs, and the resources are conceptually separate from the representations returned to the client. These resources can be manipulated using a HTTP method and a URI. Furthermore, the messages are self-descriptive. Lastly, hyperlinks are used as the engine of application state.

Milestone 6: *Share with us some queries (at least 3) in your application that require database access. Provide the* actual SQL queries *you use (if you are using an [ORM](), find out the underlying query and provide both the ORM query and the underlying SQL query). Explain what the query is supposed to be doing.*

Query 1:

```
SELECT req_id, problem_type, title, description, career_type, date_created, should_
display, name, photo_url, nus_email, matric_date, grad_date, major, telegram
FROM Requests INNER JOIN UsersInfo ON (UsersInfo.user_id = Requests.mentee_id)
ORDER BY date_created DESC
```

This query retrieves all mentorship requests of current users together with their associated information. The requests are ordered in descending order of date, which means that the latest request will be first.

Query 2:

```
SELECT DISTINCT U.name, N.nid, R.mentee_id , N.notif_type, N.date_created,
  N.is_read, R.title, R.req_id
  FROM UsersInfo U,
       (Notifies N NATURAL JOIN Mentorship M) JOIN Requests R USING (req_id)
  WHERE M.mentor_id = U.user_id
  AND R.mentee_id = $1
```

This query retrieves the information associated with notifications for a particular user. This is a parameterized query where $1 takes the value of the user's ID. The user ID is passed to the server using a HTTP POST Method.

Query 3

```
SELECT DISTINCT C.career_type, count(*) as num_of_mentions
     FROM CareerTypes C, Requests R
     WHERE C.career_type = ANY(R.career_type)
     GROUP BY C.career_type
     ORDER BY num_of_mentions DESC
```

This query retrieves the career types and the corresponding number of times that it has been mentioned by mentees. They are also ordered in descending order of number of mentions.

More SQL queries:

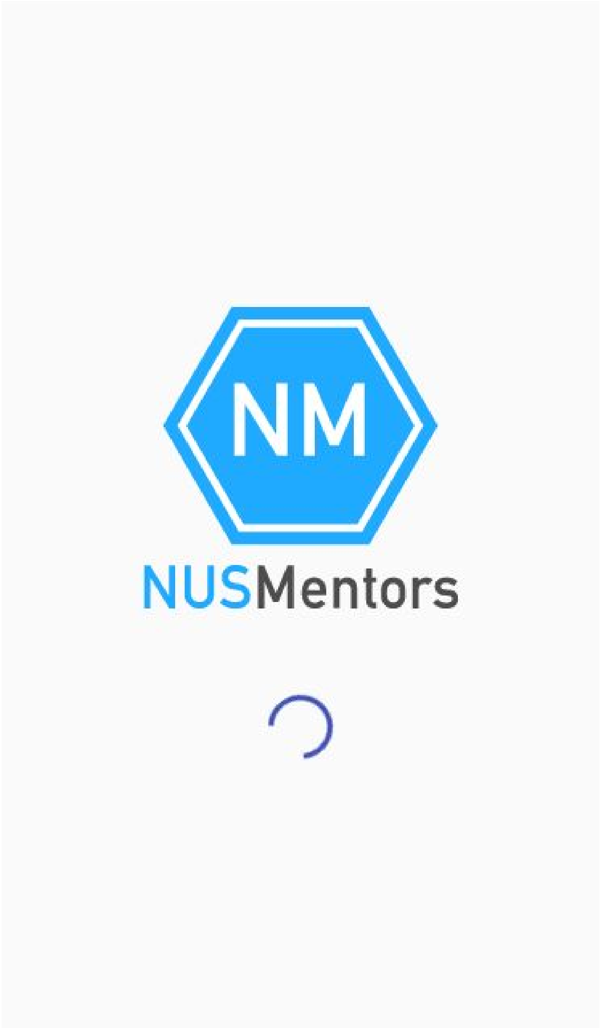
We used SQL (parameterized for security reasons) for querying the database for NUSMentors:

More SQL queries can be found in our codebase here:
https://github.com/cs3216/2020-a3-mobile-cloud-2020-group-9/api/routes/impl/

We also made use of transactions, which can be found in
https://github.com/cs3216/2020-a3-mobile-cloud-2020-group-9/api/database/procedures.sql

Milestone 7: *Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly. Include an image of the icon and a screenshot of the splash screen in your writeup. If you did not implement a splash screen, justify your decision with a short paragraph. Add your application to the home screen to make sure that they are working properly. Make sure at least Safari on iOS and Chrome on Android are supported.*

Here it is!



Milestone 8: *Style different UI components within the application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application. Choose one of the CSS methodologies (or others if you know of them) and implement it in your application. Justify your choice of methodology.*

We used CSS-in-JS for styling our application. This allowed our CSS to be scoped locally to the file, which mitigates class name conflicts. It also allows the CSS and JS to be co-located, which makes the code easier to reason with.

Our UI design reflects the design principles of the application. It is simple and draws attention to the primary actions we want the user to take. Our UI design avoids distracting features and allows the user to focus attention on the key actions, whether it is getting started, requesting for help or browsing the list of available help. We removed any unnecessary steps so that the user can efficiently complete his goal.

We used Objected Oriented CSS (OOCSS) as our CSS methodology. This allowed us more flexibility with naming and folder structure as compared to alternatives such as BEM. We thought that flexibility was important for a project which had very limited time, and so we chose OOCSS.

Milestone 9: *Set up HTTPS for your application, and also redirect users to the `https://` version if the user tries to access your site via `http://`. HTTPS doesn't automatically make your end-to-end communication secure. List 3 best practices for adopting HTTPS for your application.*

Here are 3 best practices we should follow when adopting HTTPS in our applications:

1. Get the security certificate from a reliable Certificate Authority that provides technical support. Also, choose a 2048-bit key instead of a 1024-bit key to ensure a high level of security.
2. Support HTTP Strict Transport Security, which tells the browser to request HTTPS page automatically even if the user enters http in the browser location bar. Redirect the users and search engines to the HTTPS page or resource with server side 301 HTTP redirects.
3. Do not mix security elements by loading resources using a HTTP connection. This degrades the security and user experience of the site. Embed only HTTPS content on a HTTPS page.

Milestone 10: *Implement and briefly describe the offline functionality of your application. Explain why the offline functionality of your application fits users' expectations. Implement and explain how you will keep your client synchronised with the server if your application is being used offline. Elaborate on the cases you have taken into consideration and how they will be handled.*

NUSMentors uses redux-persist to persist the store so that users can continue to use some functionalities even when they are offline.

Firstly, mentors can view the contact information of mentees that they are mentoring even when offline. This means that mentors can refer to the mentee's contact information when they are moving to a scheduled meeting location. They need not worry about losing the contact information due to poor internet connection. Similarly, the mentee can view the contact information of mentors when offline.

Secondly, mentors and mentees can view the mentorship requests even when offline. This means that the mentor can read through what the mentee needs help with. We understand that mentors are often busy with their work, so this allows them to read through the mentee's request in pockets of free time. This is a convenient feature that can allow mentors to have a better understanding of the mentee's request.

Finally, users can browse the full list of available mentorship requests when offline. This list will not be synchronized with the server if the application is offline, but it will be updated whenever the client gets internet connection. Although the list is not guaranteed to be updated, the list is not expected to change over short durations (e.g. a short bus ride home). Mentors will still find it convenient to read through these requests to see if any of them catches their interest.

Milestone 11: Compare the advantages and disadvantages of token-based authentication against session-based authentication. Justify why your choice of authentication scheme is the best for your application.

Session-based authentication stores the user state on the server. It stores the session data in on the server's memory and stores the session id in a cookie on the user browser. On the other hand, token-based authentication stores the user state on the client. The user data is encrypted (e.g. into a JSON Web Token) and sent as a header for every subsequent request.

Token-based authentication scales better than session-based authentication. When there is many users using the system at once, session-based authentication will require a large amount of server memory to store the session data, while token-based authentication does not.

Furthermore, session-based authentication violates the REST principle of statelessness by storing user state in the server. Introducing server-side state means that there is a need for synchronization logic. Server-side state also makes the requests more difficult to cache.

The advantage of using session-based authentication is that there is less work needed on the client-side as the heavy-lifting is done on the server side.

From a security point of view, applications that implement session-based authentication need to be aware of Cross-site request forgery attackers (CSRF). On the other hand, applications that implement token-based authentication are CSRF-safe, but they need to be aware of Cross-Site Scripting attacks (XSS).

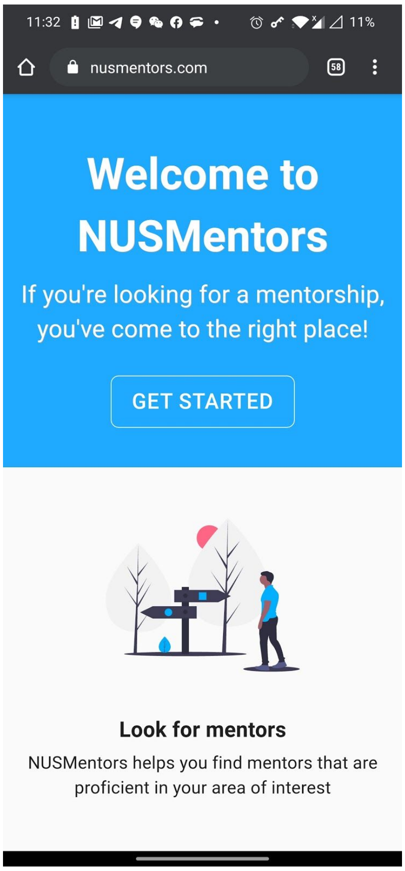
For our application, we used token-based authentication (JWTs) because it is a more scalable solution. It follows the REST principles, which enables us to remove unnecessary complexity caused by storing state on the server side. We also understand that production systems usually make use of a hybrid of both mechanisms to create flexibility and a more robust application and this is something we will explore when building production systems in the future.

Milestone 12: *Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. Lastly, list down some (at least 5) of the mobile site design principles and which pages/screens demonstrate them.*

We used material UI as our CSS framework. Material UI provides responsive components for us. The CSS frameworks mainly differ in terms of look and feel. One popular alternative framework is Bootstrap. Both bootstrap and Material UI provide responsive components using a 12 point grid.

However, material UI was designed with mobile as one of the main considerations. This is why our group felt that Material UI was the right tool for the job.

1. Keep call to actions front and center
2. Keep menus short and sweet
3. Make it easy to get back to the homepage
4. Minimize form errors with labelling and real-time validation
5. Optimize your entire site for mobile

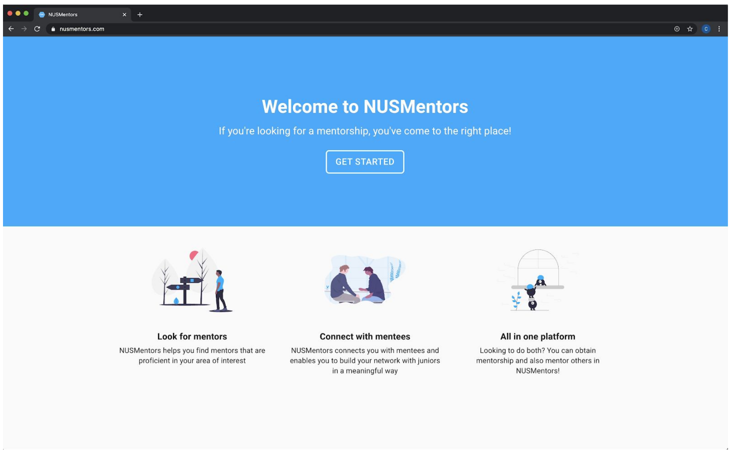


Welcome to NUSMentors

If you're looking for a mentorship, you've come to the right place!

Look for mentors

NUSMentors helps you find mentors that are proficient in your area of interest



Welcome to NUSMentors

If you're looking for a mentorship, you've come to the right place!

Look for mentors

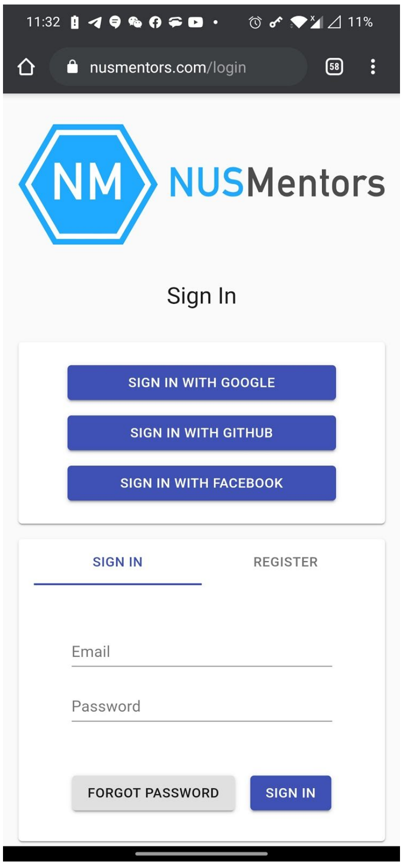
NUSMentors helps you find mentors that are proficient in your area of interest

Connect with mentees

NUSMentors connects you with mentees and enables you to build your network with juniors in a meaningful way

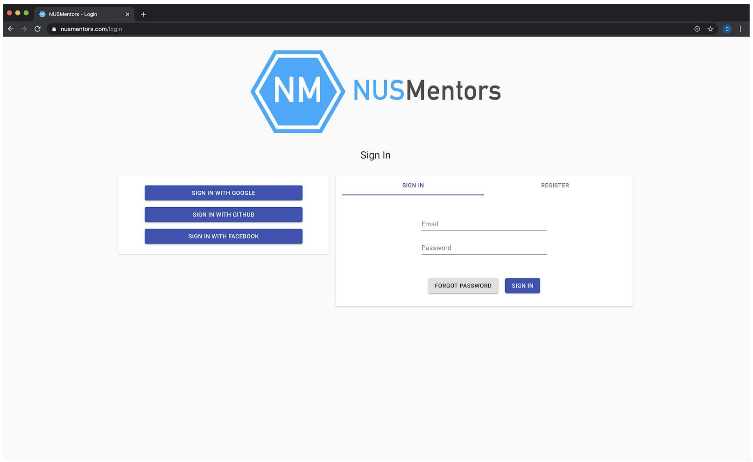
All in one platform

Looking to do both? You can obtain mentorship and also mentor others in NUSMentors!



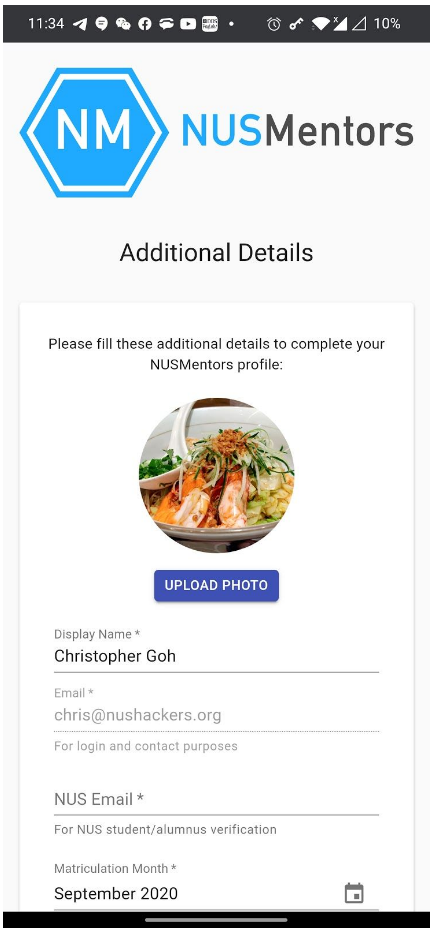
Email

Password



Email

Password



Additional Details

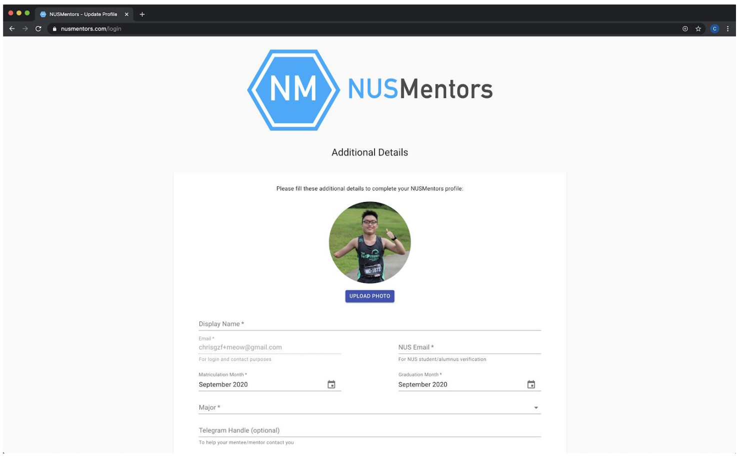
Please fill these additional details to complete your NUSMentors profile:

Display Name *
Christopher Goh

Email *
chris@nushackers.org
For login and contact purposes

NUS Email *
For NUS student/alumnus verification

Matriculation Month *
September 2020



Additional Details

Please fill these additional details to complete your NUSMentors profile:

Display Name *

Email *
chrisgzf+meow@gmail.com
For login and contact purposes

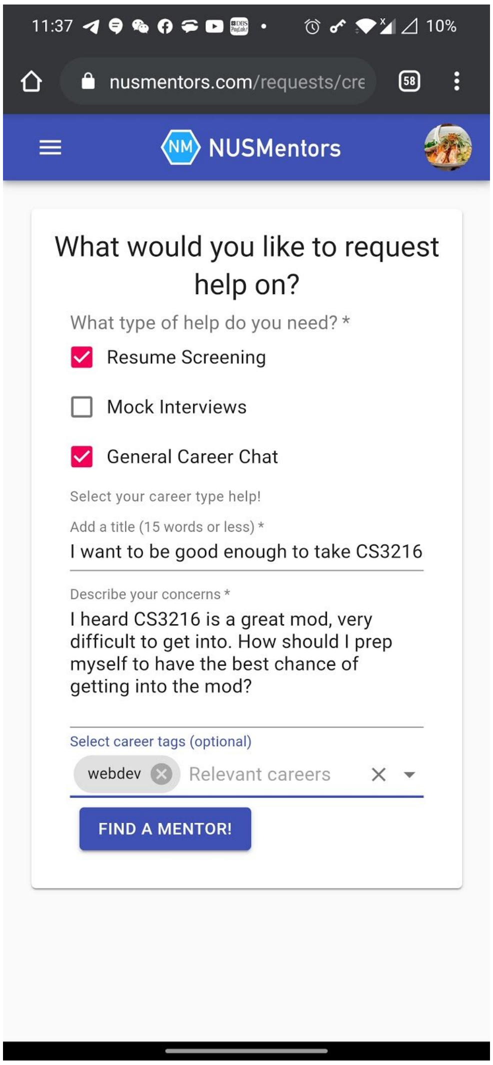
NUS Email *
For NUS student/alumnus verification

Matriculation Month *
September 2020

Graduation Month *
September 2020

Major *

Telegram Handle (optional)
To help your mentee/mentor contact you



What would you like to request help on?

What type of help do you need? *

☑ Resume Screening

□ Mock Interviews

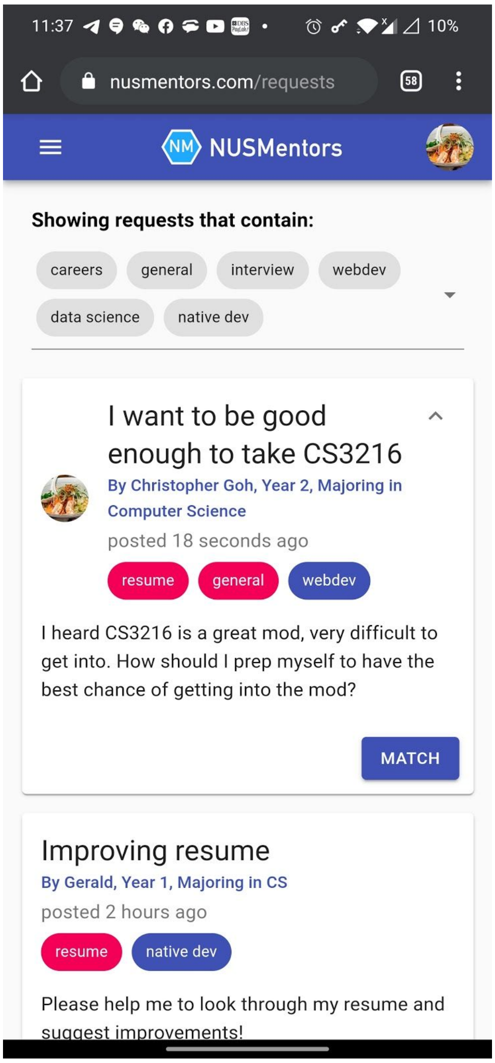
☑ General Career Chat

Select your career type help!

Add a title (15 words or less) *
I want to be good enough to take CS3216

Describe your concerns *
I heard CS3216 is a great mod, very difficult to get into. How should I prep myself to have the best chance of getting into the mod?

Select career tags (optional)
webdev ⊗ Relevant careers ✕



Showing requests that contain:

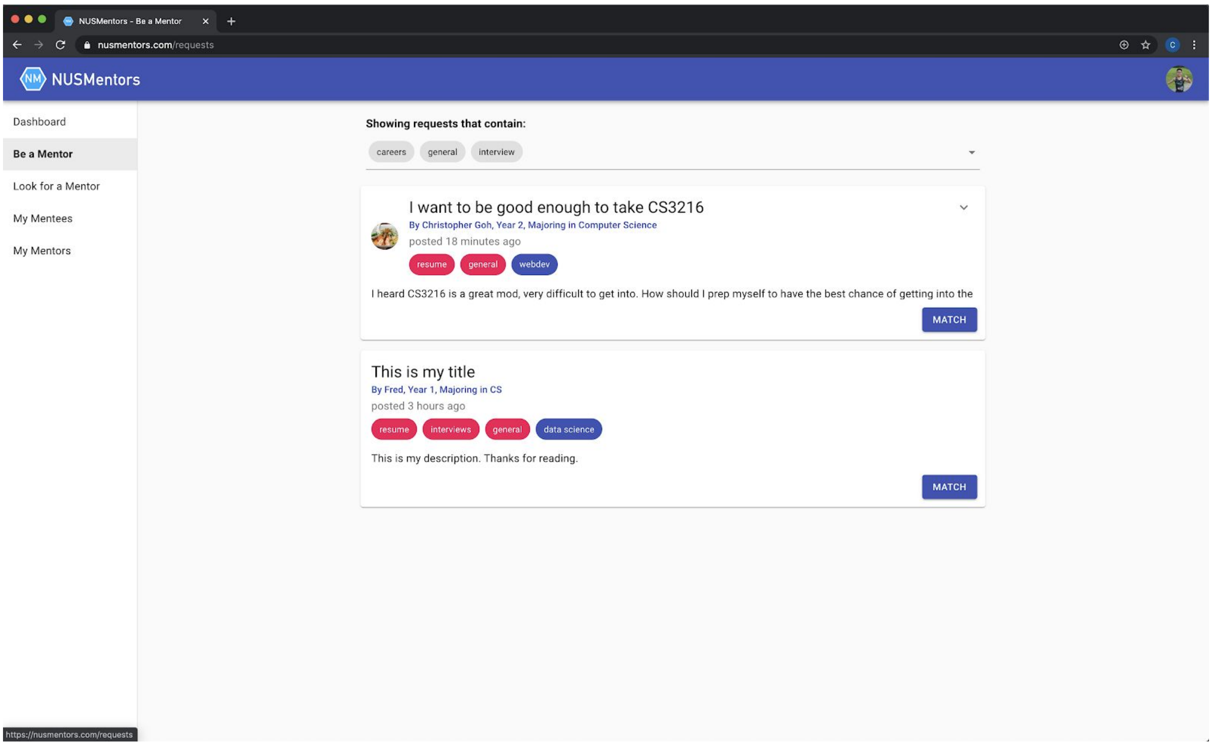
careers general interview webdev data science native dev

I want to be good enough to take CS3216
By Christopher Goh, Year 2, Majoring in Computer Science
posted 18 seconds ago
resume general webdev

I heard CS3216 is a great mod, very difficult to get into. How should I prep myself to have the best chance of getting into the mod?

Improving resume
By Gerald, Year 1, Majoring in CS
posted 2 hours ago
resume native dev

Please help me to look through my resume and suggest improvements!



Showing requests that contain:

careers general interview

I want to be good enough to take CS3216
By Christopher Goh, Year 2, Majoring in Computer Science
posted 18 minutes ago
resume general webdev

I heard CS3216 is a great mod, very difficult to get into. How should I prep myself to have the best chance of getting into the

This is my title
By Fred, Year 1, Majoring in CS
posted 3 hours ago
resume interviews general data science

This is my description. Thanks for reading.

Milestone 13: Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

1. **Onboarding the user**

For onboarding the users, we start by requesting for the contact information that they want to share with their future mentor/mentee. Regarding their site preferences, we had two options:

A. Ask the user to input their preferences of what types of requests they would like to see (so that we can filter automatically)
B. Skip (A) and go straight to the application.

We chose implementation (B), our design principle was “Keep it Simple”. Users should be able to log in and do exactly what they want. For students, they should be able to submit a request for mentorship. For mentors, they should be able to browse the list of mentorship requests. Nothing more complicated than that.

2. **Submitting a request**

We considered two possible implementations:

A. Submitting the request into a pool of available requests which mentors can choose from
B. Submitting the request to a particular mentor

We chose implementation (A), our design principle was “Pull, don’t push”. Mentors should not receive any type of notification when students request for mentorship. Instead, mentors should be pulling from the list of available mentorships and accepting mentees when they want to. This way, mentors will not be annoyed by an endless stream of notifications and instead provide high-quality mentorships because they are themselves actively seeking out mentees to help.

3. **Connecting mentors with mentees**

We considered two possible implementations:

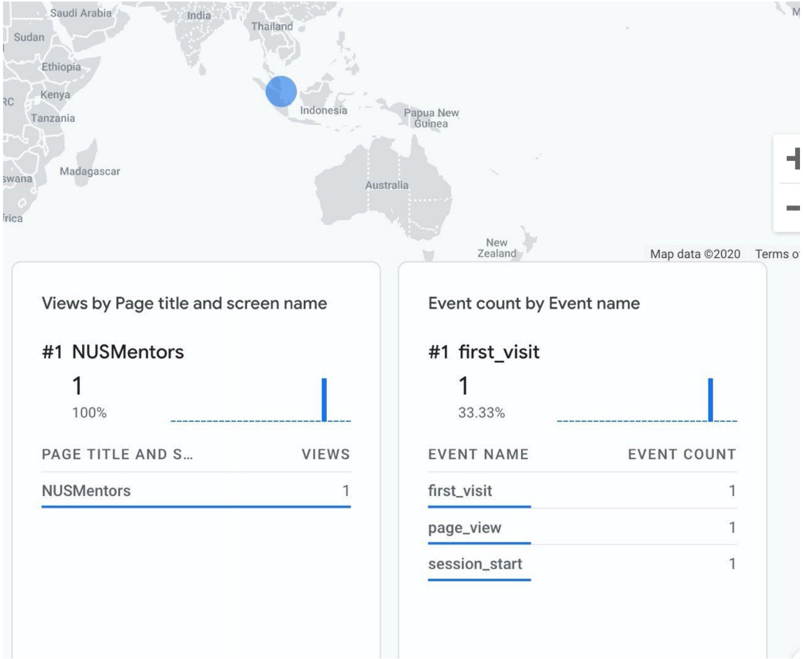
A. Provide the list of contact information for each user to the other party
B. Become the platform for further interactions by integrating chat and video functionalities

We chose implementation (A), our design principle was “Don’t ship half-baked features”. We knew that it would be useful for the platform to allow further interactions via chat or video functionalities if and only if these functionalities were on par with alternatives like Telegram or Zoom. However, we decided that it would not be feasible to implement these chat/call functionalities in such a short time. So, we made a practical decision to make NUSMentors agnostic of chat/video call platform.

In NUSMentors, users can specify how they want to be contacted via their contact information. After being matched for a mentorship, both the mentors and mentees can refer to the contact information for further contact with the other party. The advantage of this implementation is that both parties will share their most frequently used communication platform. This encourages a more natural and long-term interaction between mentor and mentee.

Milestone 14: *Embed Google Analytics in your application and give us a screenshot of the report. Make sure you embed the tracker at least 48 hours before submission deadline as updates are reported once per day.*

Completed! Here is a screenshot of the report:



Views by Page title and screen name

#1 NUSMentors

1

100%

PAGE TITLE AND S...	VIEWS
NUSMentors	1

Event count by Event name

#1 first_visit

1

33.33%

EVENT NAME	EVENT COUNT
first_visit	1
page_view	1
session_start	1

Milestone 15: Achieve a score of at least 12/14 for the Progressive Web App category and include the Lighthouse html report in your repository.

Completed too! Here is the lighthouse html report:

https://github.com/cs3216/2020-a3-mobile-cloud-2020-group-9

Milestone 16: *Identify and integrate with social network(s) containing users in your target audience. State the social plugins you have used. Explain your choice of social network(s) and plugins. (Optional)*

We’ve integrated Facebook, GitHub, and Google social login, which allows us to use their social media display name as their in-app name in NUSMentors, and hence be a one-click registration.