

Schoolines

CS3216 Assignment 3 Milestones

Ng Chuen Weng Eugene | Zhang Hanming | Ken Oung Yong Quan | Ten Zhi Yang

Milestone 0

Describe the problem your application solves.

Every student faces countless projects, assignments and examinations throughout their academic years, and with these come a dreadfully long list of deadlines to meet. Keeping up with all these deadlines is a chore.

The more organized students use diaries, to-do lists, Google Calendar or some kind of specialised tool to keep track of everything. But this is a lot of work, so the less driven students may just wait for their friends to remind them of the deadlines.

Anecdotal Evidence: An oft-heard conversation in NUS.

Student A: "Have you submitted Assignment X for CSXXXX? Can I take a look?"

Student B: "Wait what assignment?!"

Student A: "Erm...the one due in 2 hours?"

Student B: "FML"

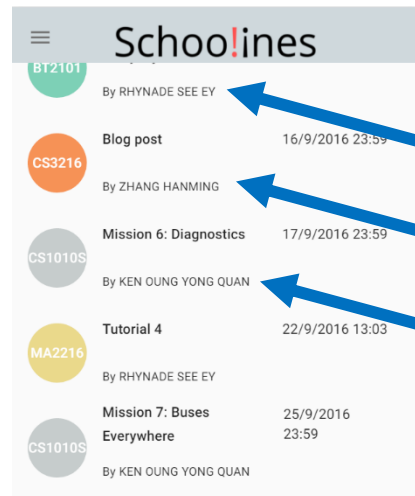
The problem that students face is that there is no system that organizes all the deadlines for them. While it is possible for students to do it themselves, a significant number of students do not bother and thus suffer when deadlines get closer.

Our solution? Schoolines is an application that allows students to benefit from the effort of the community through the consolidation of all their deadlines into one list. With Schoolines, the effort of a few students to submit deadlines will be able to benefit the entire class. In exchange for their time and effort, the student will be credited which will earn him some street cred.

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?

In a nutshell, Schoolines is a web application that consolidates submission deadlines as well as exam dates through crowdsourcing. We believe this will be incredibly effective since information is non-rivalrous in consumption. With only one student's contribution, the entire class can benefit from the information he or she shares. To encourage such contribution, we credit the contributor of each deadline. This also provides a check against fake or spammy deadlines, because then everyone will know that you're a terrible person with no morals.



Why does this make sense as a mobile application? The point of this app is to allow students, even the lazy ones, to have all their deadlines at their fingertips. And all of us know that students cannot part with their mobile phones. If we want students to have the easiest access to all their deadlines, it is clear that the solution would be to provide them this information via their phones. Furthermore, the offline functionality means that students will still be able to access their deadlines even when they're trying to conserve their mobile data and the NUS WiFi is spotty. They would only need to use the app online every so often for the deadlines to be synced with the server.

Milestone 2

Describe your target users. Explain how you plan to promote your application to attract your target users.

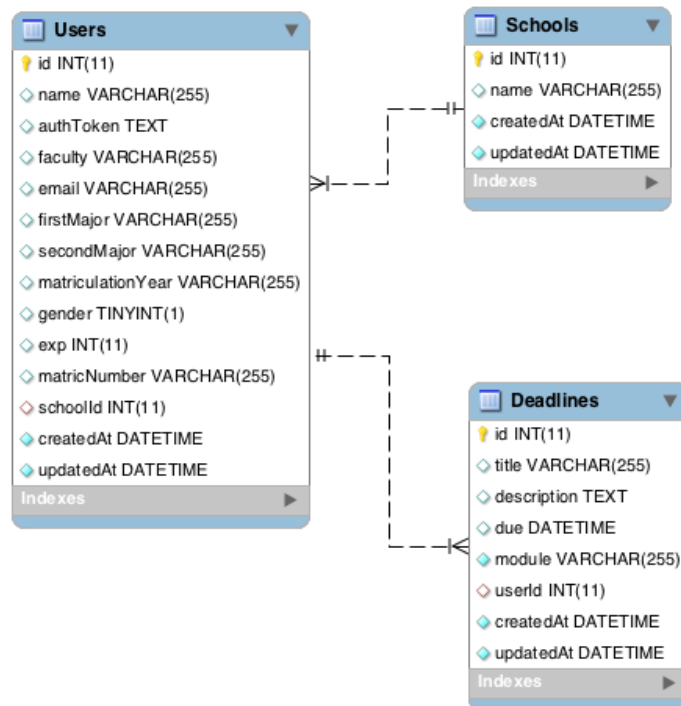
Our main target users are university students.

We plan to promote our application through channels that university students refer to for academic information. We could link up with faculties, clubs, and university student life departments, as well as collaborate with reliable online academic planning solutions such as NUSmods. The touch points for awareness would be faculty facebook pages, email blast and freshmen orientation academic talks.

Furthermore, we could conduct competitions every semester to identify one user from each faculty that contributed the most number of accurate deadlines. This will encourage users to fill up deadlines which can benefit the NUS community as a whole and at the same time spread the word about Schoolines.

Milestone 3

Draw the database schema of your application.



Clarification: We added the table "Schools" to accommodate future expansion plans. For now, our application will be completely based in NUS, so the table "Schools" is not in use.

Milestone 4

Design and document all your REST API. The documentation should describe the requests in terms of the triplet mentioned above. Do provide us with a brief 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.)

The API has uniform interface whereby data are passed between servers and clients in JSON only. Also the clients and server are separate from each other as the server does not keep a session for each client. All the necessary information and data necessary for communications are contained within the url or query parameters. The clients in our app is written with AngularJS and all the requests to server are sent using Angular's \$http service which uses AJAX call.

| Create User | |
|------------------|--|
| URL | /userManagement/createUser |
| Method | POST |
| URL Params | None |
| Data Params | { token=[TEXT] } |
| Success Response | <p>Code: 201 Content: { userId: 1, status: "success" }</p> <p>OR</p> <p>Code: 201 Content: { userId: 1, status: "exist" }</p> |
| Error Response | <p>Code: 400 Content: { Status: "failed" }</p> |
| Sample Call | <pre>\$ajax({ url: "userManagement/createUser", datatype: "json",</pre> |

| | |
|-------------|---|
| | <pre> type: "POST", data: { token: "SDFKJWEROIUSDFLKSJFD" }, success: function(res){ console.log(res); } }); </pre> |
| Description | Use IVLE token to retrieve user's information from IVLE API and store them in the database. |

| Create Deadline | |
|------------------|---|
| URL | /deadlineManagement/create |
| Method | POST |
| URL Params | None |
| Data Params | <pre> { deadline={ userId=[INTEGER], token=[TEXT] } } </pre> |
| Success Response | Code: 201 Content: None |
| Error Response | Code: 400 Content: None |
| Sample Call | <pre> \$.ajax({ url: "deadlineManagement/create", datatype: "json", type: "POST", data: { deadline: { userId: 1, token: "ALSDKFJSDFOSWEIRJLS" } }, success: function(res){ console.log(res); } }); </pre> |

| | |
|-------------|---|
| Description | Create a deadline with the contributor being the user associated with userId and the corresponding token. Return 400 when userId does not match the token stored in the database. |
|-------------|---|

| Get Modules | |
|------------------|---|
| URL | /userManagement/getModules |
| Method | POST |
| URL Params | None |
| Data Params | <pre>{ token=[TEXT] }</pre> |
| Success Response | <p>Code: 200</p> <p>Content: { moduleCodes: ["module1", "module2", "module3"], status: "success" } </p> |
| Error Response | <p>Code: 400</p> <p>Content: { status: "failed" } </p> |
| Sample Call | <pre>\$.ajax({ url: "userManagement/getModules", datatype: "json", type: "POST", data: { token: "SDFKJWEROIUSDFLKSJFD" }, success: function(res){ console.log(res); } });</pre> |
| Description | Get the modules that the user is currently taking through IVLE APIs |

| Get Deadlines | |
|------------------|--|
| URL | /deadlineManagement/getDeadlines/ |
| Method | GET |
| URL Params | 0=[STRING] 1=[STRING] 2=[STRING] ... |
| Data Params | None |
| Success Response | Code: 200 Content: { deadlineArray: [deadline1=[JSON], deadline2=[JSON], deadline3=[JSON], ...] } |
| Error Response | None |
| Sample Call | \$.ajax({ url: "deadlineManagement/getDeadlines/?0='CS1010'&1='CS1231'", datatype: "json", type: "GET", success: function(res){ console.log(res); } }); |
| Description | Get all the deadlines associated with the requested modules in the database |

| Delete Deadline | |
|------------------|--|
| URL | /deadlineManagement/delete |
| Method | POST |
| URL Params | None |
| Data Params | <pre>{ id=[INTEGER], token=[TEXT] }</pre> |
| Success Response | Code: 200 Content: None |
| Error Response | Code: 401 Content: None OR Code: 400 Content: None |
| Sample Call | <pre>\$.ajax({ url: "deadlineManagement/delete", datatype: "json", type: "POST", data: { id: 1, token: "SDFKJWEROIUSDFLKSJFD" }, success: function(res){ console.log(res); } });</pre> |
| Description | Delete the deadline associated with the requested id authorized by the token |

Milestone 5

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 explain how it works.

Select all the Deadlines with associated Users where the deadlines' module matches one of the modules given and return the retrieved Deadlines sorted by the date of the deadline chronologically.

```
SELECT `Deadline`.`id`, `Deadline`.`title`, `Deadline`.`description`, `Deadline`  
`.`module`, `Deadline`.`due`, `Deadline`.`createdAt`, `Deadline`.`updatedAt`,  
`Deadline`.`userId`, `  
  
Deadline`.`userId`, `User`.`id` AS `User.id`, `User`.`name` AS `User.name`,  
`User`.`authToken` AS `U  
  
ser.authToken`, `User`.`faculty` AS `User.faculty`, `User`.`email` AS `User.email`,  
`User`.`firstMaj  
  
or` AS `User.firstMajor`, `User`.`secondMajor` AS `User.secondMajor`,  
`User`.`matriculationYear` AS  
  
`User.matriculationYear`, `User`.`gender` AS `User.gender`, `User`.`exp` AS  
`User.exp`, `User`.`matr  
  
icNumber` AS `User.matricNumber`, `User`.`createdAt` AS `User.createdAt`,  
`User`.`updatedAt` AS `Use  
  
r.updatedAt`, `User`.`SchoolId` AS `User.SchoolId`, `User`.`schoolId` AS  
`User.schoolId` FROM `Deadl  
  
ines` AS `Deadline` LEFT OUTER JOIN `Users` AS `User` ON `Deadline`.`userId` =  
`User`.`id` WHERE `De  
  
adline`.`module` IN ('CFG1010', 'CS2105', 'CS2106', 'CS2108', 'CS3216', 'CS3241')  
ORDER BY `Deadline  
  
`.`due` ASC;
```

Store user's information retrieved from IVLE to database.

```
INSERT INTO `Users` (`id`,`name`,`authToken`,`faculty`,`email`,`firstMajor`,`secondMajor`,`matriculationYear`,`gender`,`exp`,`matricNumber`,`createdAt`,`updatedAt`)
VALUES (DEFAULT,'ZHANG
HANMING','746D9BB5A7F11E6C8F251862B66A07F1D00285370BBA1CAC853FADB157963349CAEBF54B6F8
4F07D5CB76B54EAD766CA4076FF279BF0106C456CE1A337ED7EF2B891E6875822BB928EDBFE19FD34B76C
4B6FC46E05F97F36F205008C5A1B003E593988FBD19EFC18340EE807E43DE7849179EED9292861AFCE0A6
38FD32874F700EB307C71B6552817B0C97AF1071DFE4B3AF6FEF10A1F2463FC349A2447E15A6DC8EF01F0
2B2325499AE50E04A631A693AED516EF0B4469F0C9699E14D58CDDA4C8ACC20113C1BE07F95C6E40C51D6
98C50A2D17DD0E60D6AD790E9B721B11B95141056B49FA5DB1914CEEF393395'

,'School of Computing','e0005541@u.nus.edu','Computer Science
(Hons)', '', '2015', true, 0, 'e0005542', '2
016-09-22 08:52:57', '2016-09-22 08:52:57');
```

Store the deadline created by a user into the database.

```
INSERT INTO `Deadlines` (`id`,`title`,`module`,`due`,`createdAt`,`updatedAt`,`user
setId`) VALUES (DEFAULT, 'A', 'CS3216', '2016-09-22 09:01:16', '2016-09-22
09:01:24', '2016-09-22 09:01:24', '1');
```

Milestone 6

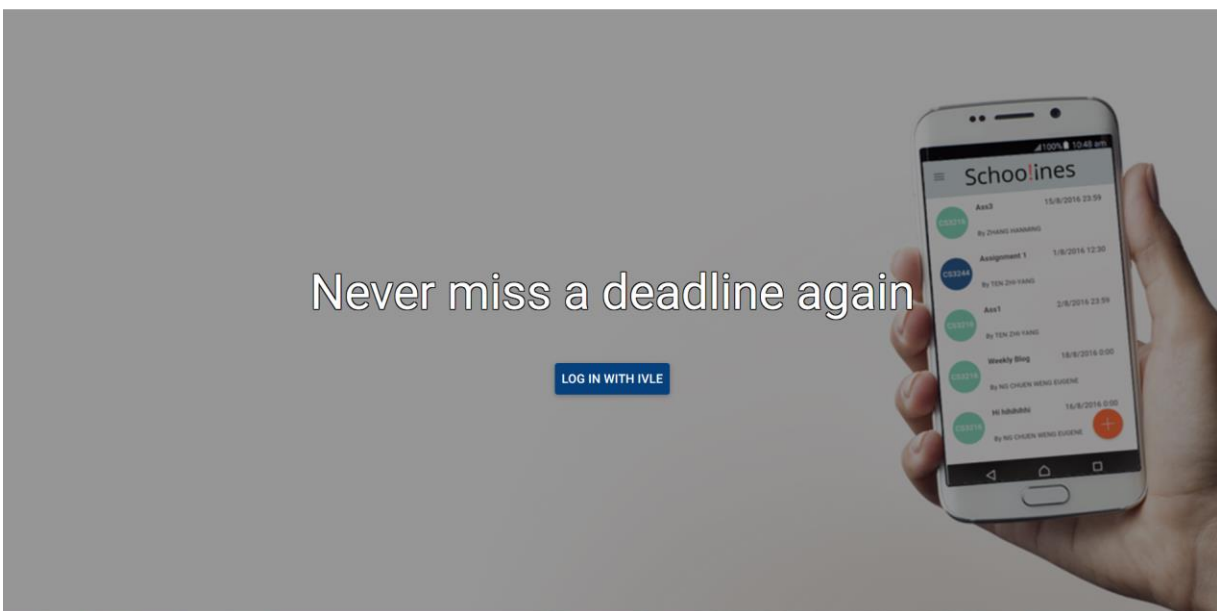
Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly. Include an image of the icon and a screenshot of the splash screen in your writeup. If you did not implement a splash screen, justify your decision 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.



Icon

Schoo!ines

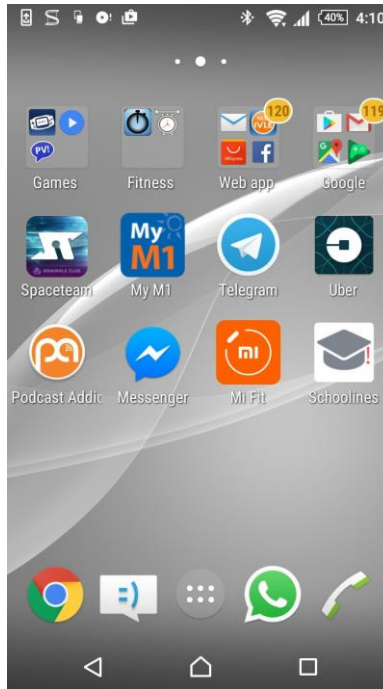
Logo



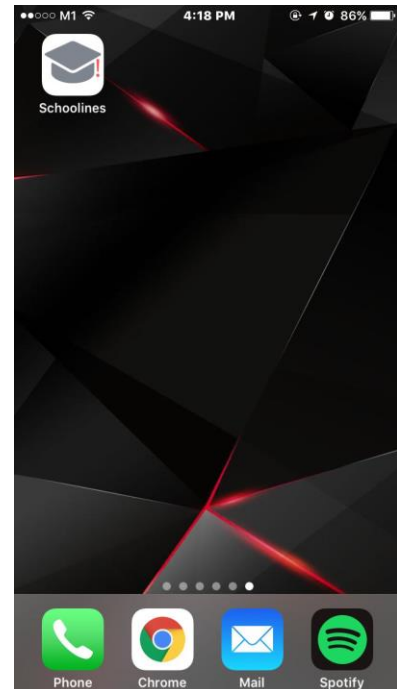
Splash page pc (width >= 1280px)



Splash page (< 1280px)



Android (Chrome)



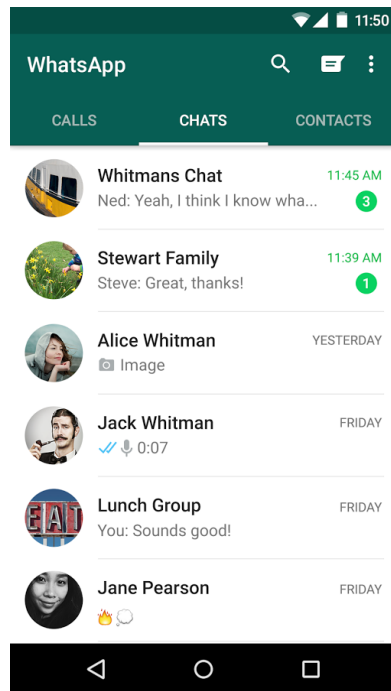
iOS (safari)

Milestone 7

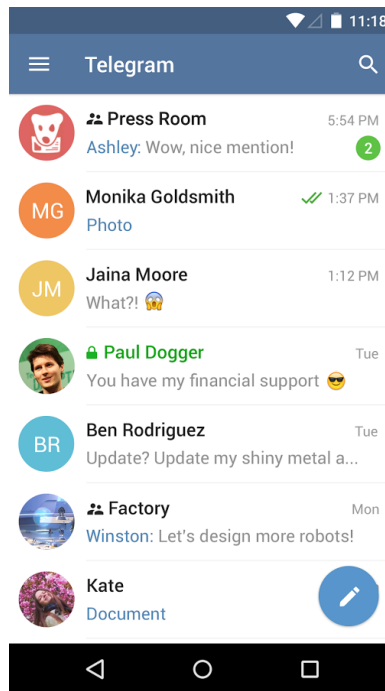
Style different UI components within the application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application. Choose one of the CSS methodologies (or others if you know of them) and implement it in your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

Our UI design is similar to Telegram and Whatsapp so that we can provide users with a sense of familiarity when they use Schoolines.

Different colors were used to distinguish different modules, with support for 10 distinct colors to represent a maximum of 10 different modules. Since the maximum workload is 27mc for concurrent degree students, the 10 distinct colors should suffice even for students taking 7-9 modules in a semester.



WhatsApp



Telegram



Schoo!ines

Our CSS methodology follows the OOCSS methodology loosely. This is similar to angular-materials' CSS practices, where the certain features are abstracted into separate class. Since the number of "views" we have are little, the consensus was that having a CSS methodology is overkill. However, by loosely following a methodology, we were able to work with much more maintainable stylesheets, where similar objects could be changed easily.

Milestone 8

Set up HTTPS for your application, and also redirect users to the https:// version if the user tries to access your site via http://. Name 3 best practices for adopting HTTPS for your application. Explain the term "certificate pinning" and discuss the pros and cons of adopting it.

1. Ensure sufficient hostname coverage

Make sure that the certificate covers all the different ways a user might visit the site. For example, if you only registered your certificate for schoolines.com, then anyone who tries to visit an alternative url such as www.schoolines.com encounters an error message similar to the one you see to the right.

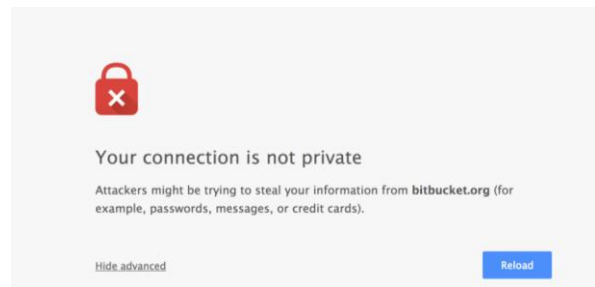


Image obtained from
http://drupaldroppings.com/sites/default/files/inline-images/google-chrome_vour_connection_is_not_private.png

2. Deploy HTTP Strict Transport Security (HSTS)

Using HSTS means that web browsers can only interact with the server via HTTPS, instead of HTTP. Even if users try to visit <http://schoolines.com>, they will automatically be redirected to the more secure <https://schoolines.com>. This helps to prevent protocol downgrade attacks and cookie hijacking.

3. Use Robust Security Certificates

Get certificate from a reputable Certificate Authority that helps to verify that you own the domain, thus preventing man-in-the-middle attacks. It would be even better if you obtained Organization Validation (OV) or Extended Validation (EV), which involves vetting of not only the domain, but also the owner of the site. While Let's Encrypt provides free certificates, those only provide Domain Validation.

You should also opt for a higher level of security by using a 2048-bit key for your certificate.

Certificate Pinning

Certificate pinning means that you only trust specific certificates that are signed by specific Certificate Authorities (CA) or match a specific set of specified public keys. If the key does not match either one of the above criteria, then it has likely been compromised.

Pros: The main benefit of performing certificate pinning is that it prevents man-in-the-middle attacks. For example, Google was able to detect the use of fraudulent certificates issued by DigiNotar using this method, and thus protect its Google Chrome users. For more information, you can refer to this post on Google's blog: <https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html>.

Cons: The problem with certificate pinning is that it assumes that the first interaction is secure. Since the certificate or public key is stored during the first encounter, it also means that attackers can still perform attacks if they are able to hijack the connection the first time.

Milestone 9

Implement and briefly describe the offline functionality of your application. Explain why the offline functionality of your application fits users' expectations. State if you have used service workers, Web Storage, or any other technology. Explain your choice. Make sure that you are able to run and use a reasonable subset of features of your application from the home screen without any internet connection.

The core feature of our app that most users will be using would be viewing and filtering of deadlines. Most of these users will also be hiding deadlines that they no longer want to see. All of these will still remain available to the user even when they are offline.

Service workers are used to cache the app shell upon their first visit to the site. Information such as the modules they take, and all deadline related data is stored in **localStorage**. Things like filtering still remain available since the processing occurs on the client-side using Angular.

The only features available offline are those related to creating new deadlines. Despite the importance of catering to "deadline contributors", these remain only a small proportion of who we hope will use the application. For a single module, you only need one or two enthusiastic persons to post all the deadlines for that module. The other hundred students taking the class will not have to do any posting. Hence, features such as creating, modifying and deleting deadlines will only be used by a small subset of our users. Therefore, we feel that it is justified for those to remain as online-only features.

Milestone 10

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.

When the application is being used offline, all interactions with the application will be made on the client-side, and the related data will be processed and stored in **localStorage**. As time goes by and the user remains offline, there could be new deadlines added to our online database. At the same time, the modules that the user takes could also have changed. All these changes will not be reflected as long as the user stays offline.

Once the user comes online, the client will make API calls to the server to update (1) modules taken and (2) list of relevant deadlines. These changes will then be reflected in the application

On the other hand, the application has been designed in such a way that the offline capabilities will not result in any need for an update in the online database. To demonstrate this, let's consider what users can do offline.

1. View deadlines and related details
2. View modules taken
3. Filter by modules
4. Hide modules

Of the four, the first three does not involve modifying any data. The fourth, hiding of modules, is implemented on the client-side, and only involves the use of **localStorage**, and will hence does not need to be synced to the online database.

Since adding or removing deadlines is not part of the application's offline functionality, only the client needs to sync with data from the server, while the server will not need to sync with client-side data.

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.

In session-based authentication, the server issues tokens to the client and clients attach the token to each requests it send to the server. Server needs to persist the state of each clients which takes up memory and computing power. However, the advantage of token-based authentication is that it reduces memory and CPU usage of the server because the server does not persist a session for each client. Hence it is more scalable than session-based authentication for the server.

Since we incorporated IVLE into our app, we can make use of the token issued by IVLE as a form of authentication. When clients make requests to the server on sensitive operations such as creating and deleting deadlines, they has to attached the token associated with the current user and the server will check if the token matches the user. This method does not take up memory and CPU of the server as the server doesn't persist a session for each client. It also makes use of the IVLE token as only users with the correct username and password for IVLE can gain access to the token.

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 have used Angular Material for the user interface. Angular Materials components follows the Google Material Design Specification. This means that anyone who uses google products, be it Google Drive or Gmail will feel familiar with our user interface.

Design with thumbs in mind

Our selectable surfaces are big, to make sure that no precise actions are required.

Simple Navigation

We wanted to make sure that the user would not have too many navigational tools, with android having a navigation bar below (in image) and safari for IOS also having a bar below.

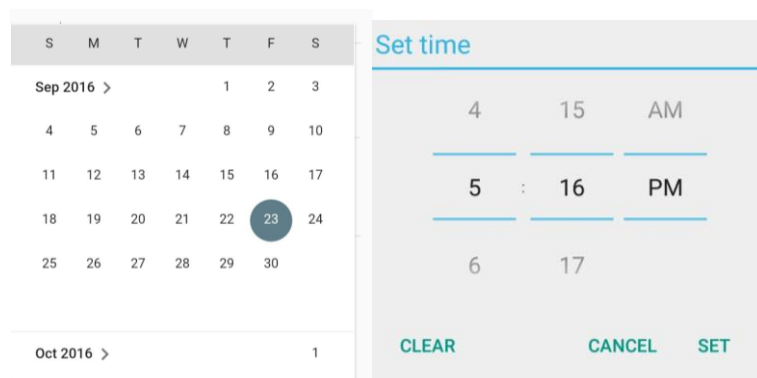
Having a responsive filter bar also allows us to give the user a more objective view

Easy to get back to home page

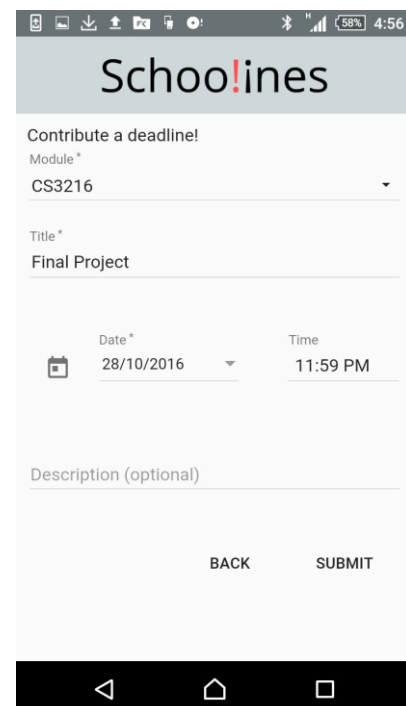
Not only is there a back button for every view in the app, the user can also navigate back to the home page by clicking the logo at the top.

Short Forms

Our form has only 4 compulsory fields, of which only the title is typed. We used date and time pickers that are familiar to the users with angular material's datepicker (google's datepicker) and the html5 timepicker, which uses the browser's time picker. This also ensures that our form will have less keystrokes required to fill out.



The image shows two screenshots of mobile UI components. On the left is a date picker for September 2016, with the 23rd highlighted. On the right is a 'Set time' dialog showing 5:16 PM, with 'AM' and 'PM' options and 'CLEAR', 'CANCEL', and 'SET' buttons at the bottom.



The screenshot shows the 'Contribute a deadline!' form in the Schoo!ines app. It includes a 'Module *' dropdown set to 'CS3216', a 'Title *' field with 'Final Project', a 'Date *' picker set to '28/10/2016', and a 'Time' picker set to '11:59 PM'. There is a 'Description (optional)' text area and 'BACK' and 'SUBMIT' buttons at the bottom. The bottom navigation bar shows back, home, and recent icons.

(Load) time is precious

We included Service workers to cache all required documents. Loading time will only feel slow for the first load, and any new versions of our program that we release. If the version is the same as the one on our server, the browser will immediately load the page from cache, making it faster and smoother for the user..



The IVLE API can be quite slow. When the user first logs in, it takes some time for us to gather all the required information from IVLE, and for us to serve them the relevant deadlines. While all this is taking place, we use a loading icon on the first page so as to entertain the user while he or she is waiting.

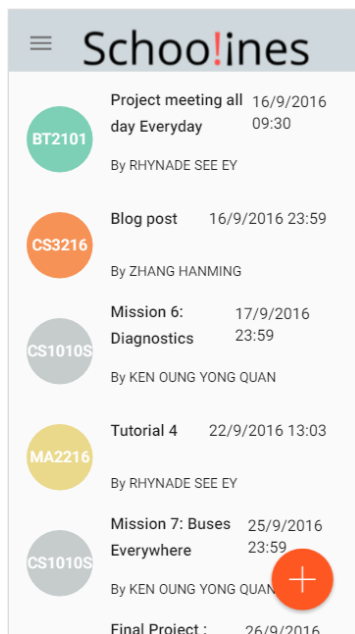
The application is also built in such a way that they will only need to login once, hence minimizing wait time.

Milestone 13

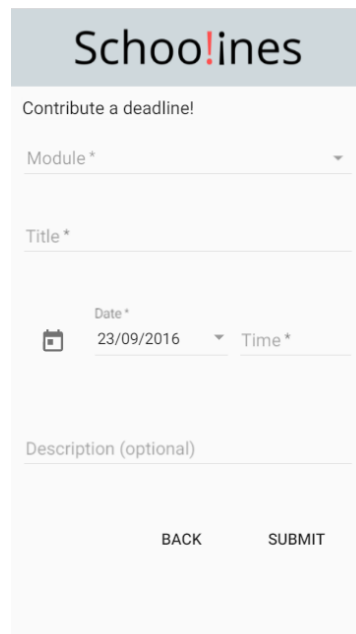
Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

The 3 common workflows within Schoolines is to “Create a deadline”, “Hide a deadline” and to “Filter deadlines based on modules”.

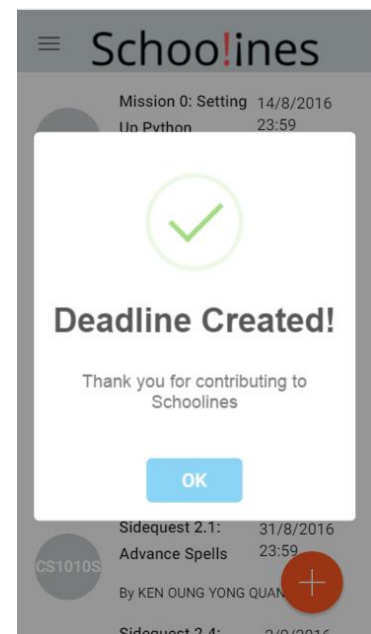
1. **Create a deadline:** Allows users to submit assignment and test deadlines. By creating a deadline, users will then be able to take note of the important academic dates. Furthermore, this process helps other users with the same module as the registered deadline will be shared across students taking the same module.



Click the ADD button



Fill in deadline details



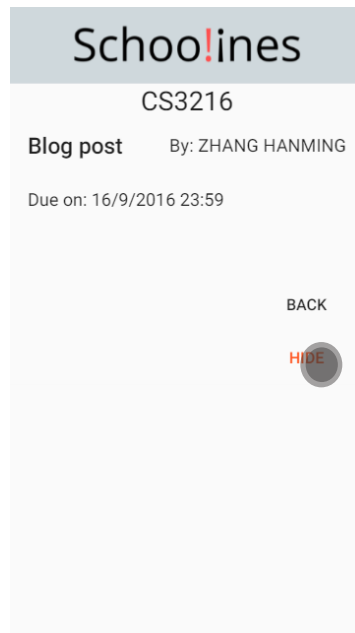
Deadline is created

We felt that this method of contributing deadlines would be the most intuitive. Instead of hiding the button in a sidebar, we use a big red 'plus' button that encourage users to tap on it and contribute a deadline.

2. **Hide:** Users are able to hide deadlines which are irrelevant to them and thus prevents their list from being overly populated with outdated information.



Select the deadline to hide



Click on hide

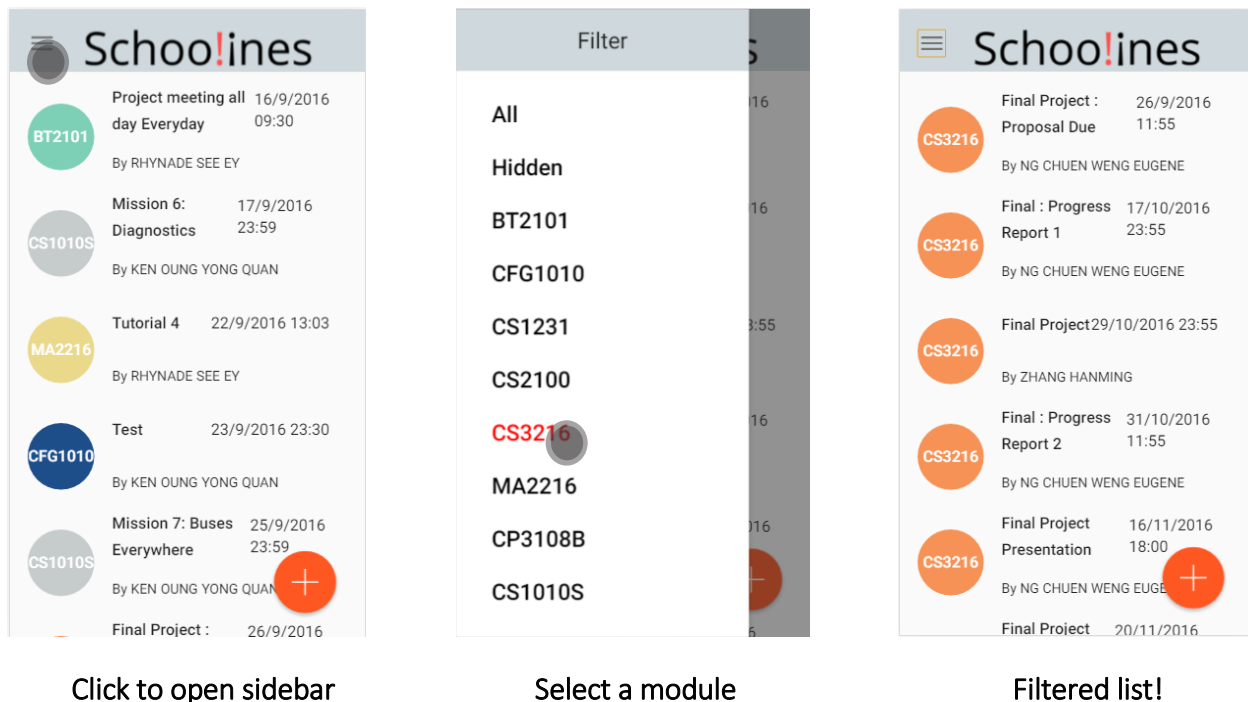


It's now hidden!

At first, we wanted to implement hiding using a simple swipe left or right. However, we soon found out that Safari on iOS uses the swipe gesture for sending users to their last visited webpage.

Given the limited real estate on the main page, we decided that we could put the hiding button inside the deadline details page so as not to clutter the main page with additional buttons.

3. **Filter deadlines:** Allow users to filter deadlines based on modules. This helps users to identify deadlines pertaining to particular module and thus make it easier to keep track of deadlines.

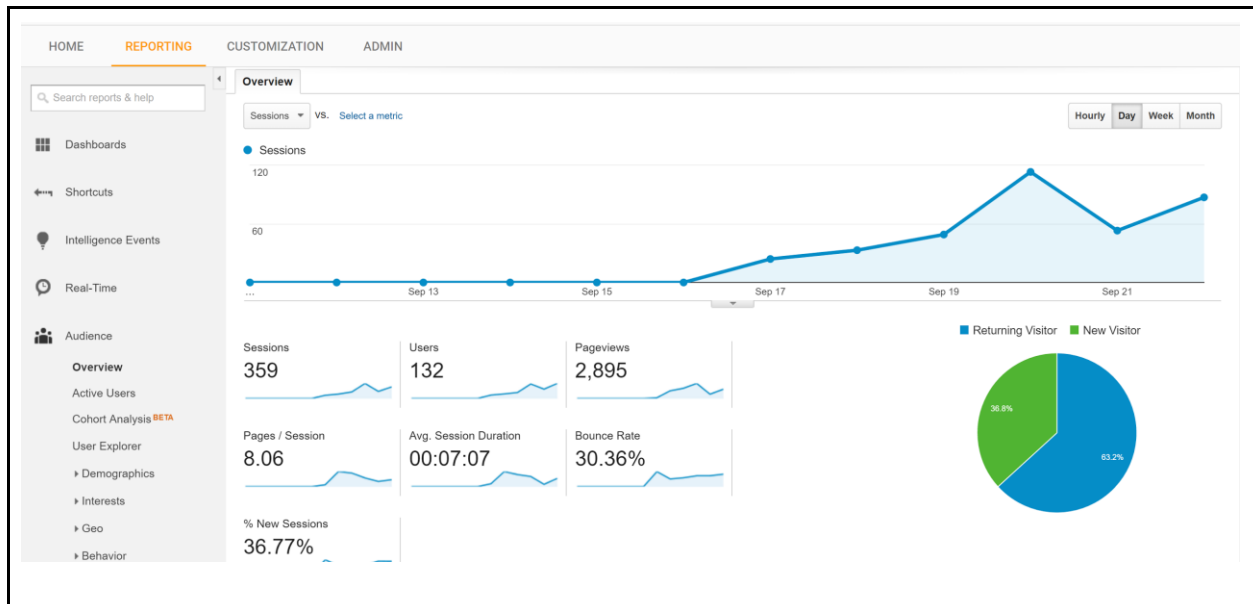


Filtering options are stored in the sidebar. Upon clicking on any module name, the sidebar closes and you see only the relevant modules. We feel that this method of doing things is rather intuitive, and easy to use.

Other methods to implement this would require stuffing more options into the main page, which we feel would cause it to be more cluttered than necessary.

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.



Milestone 15

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)

A “social network” that we’ve integrated is IVLE. Our target audience are NUS students, and using IVLE helps us retrieve information on their modules without needing the users to have a lengthy sign in process.

The "social plugin" used in this case would be the IVLE login.

That said, we do not really have any social interaction on our web application, as we find that it would enhance the experience for users. The only social element would be that people can see who contributed a certain deadline, thus boosting the contributor's social status among those taking the module.

Milestone 16: Make use of the Geolocation API in your application. (Optional)

The Geolocation API was not used.