

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, todo 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 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 did not bother and thus suffer when the deadline gets 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 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.

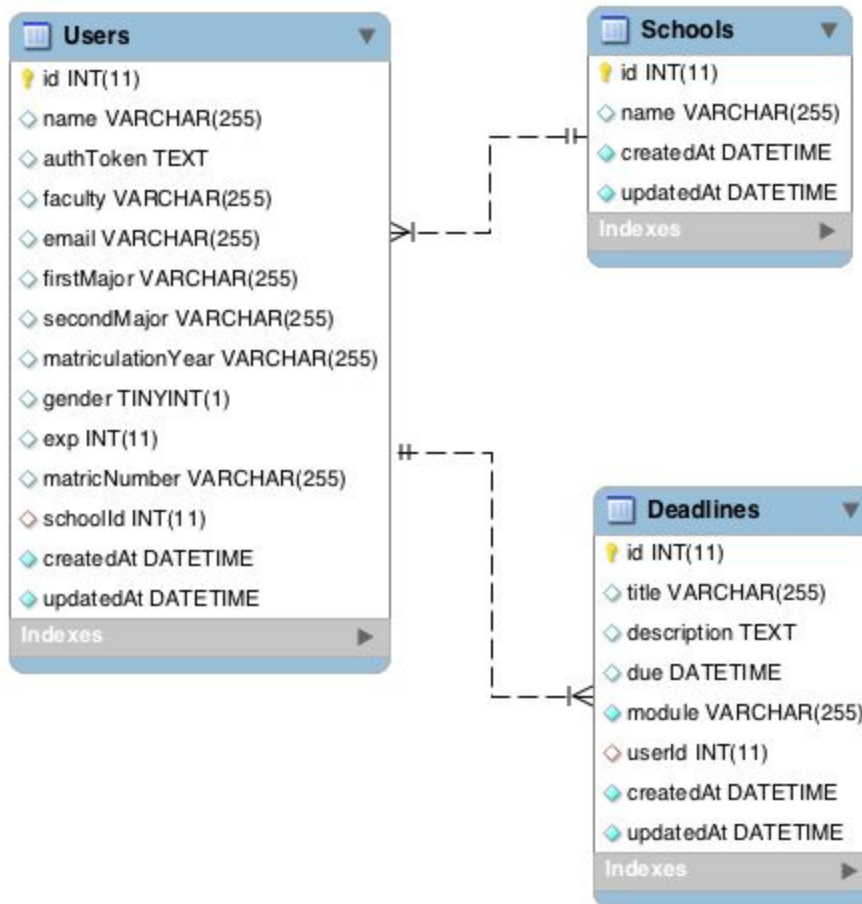
Target users: Students, Teaching Assistants and Lecturers of NUS in Phase 1, followed by other academic institutions.

Main target users: 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, university student life departments and 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” in case the app expands into other schools in the future. For now it is only operating 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.)

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

	<pre> }, 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: {</p> <pre> moduleCodes: ["module1", "module2", "module3"], status: "success" }</pre>
Error Response	<p>Code: 400</p> <p>Content: {</p> <pre> status: "failed" }</pre>
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 `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;
```

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.

```
INSERT INTO `Users` (`id`,`name`,`authToken`,`faculty`,`email`,`firstMajor`,`secondMajor`,`matriculationYear`,`gender`,`exp`,`matricNumber`,`createdAt`,`updatedAt`) VALUES (DEFAULT,'ZHANG HANMING','746D9BB5A7F11E6C8F251862B66A07F1D00285370BBA1CAC853FADB157963349CAEBF54B6F84F07D5CB76B54EAD766CA4076FF279BF0106C456CE1A337ED7EF2B891E6875822BB928EDBFE19FD34B76C4B6FC46E05F97F36F205008C5A1B003E593988FBD19EFC18340EE807E43DE7849179EED9292861AFCE0A638FD32874F700EB307C71B6552817B0C97AF1071DFE4B3AF6FEF10A1F2463FC349A2447E15A6DC8EF01F02B2325499AE50E04A631A693AED516EF0B4469F0C9699E14D58CDDA4C8ACC20113C1BE07F95C6E40C51D698C50A2D17DD0E60D6AD790E9B721B11B95141056B49FA5DB1914CEE F393395','School of Computing','e0005541@u.nus.edu','Computer Science (Hons)',",','2015',true,0,'e0005542','2016-09-22 08:52:57','2016-09-22 08:52:57');
```

Store user's information retrieved from IVLE to database.

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

Store the deadline created by a user into the database.

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.



Schoo!ines

Icon

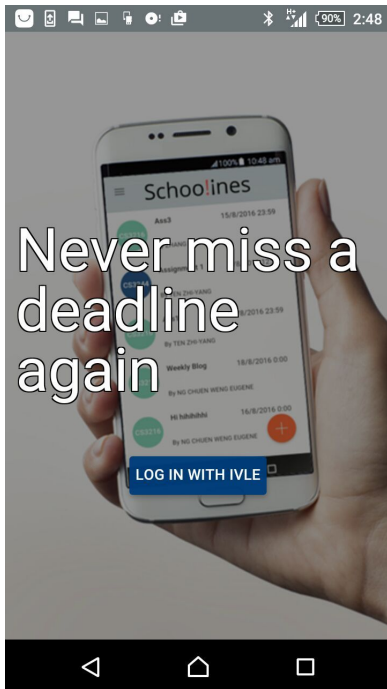
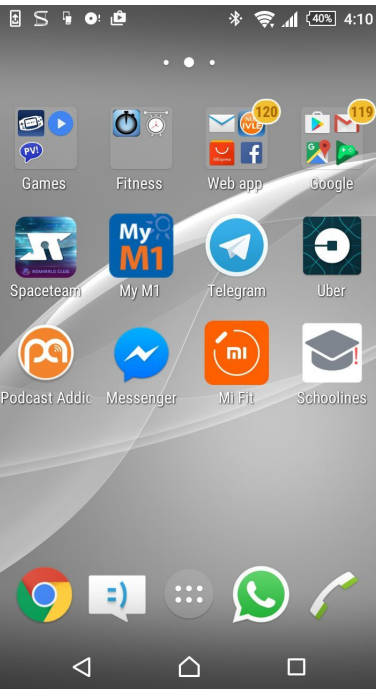

Logo

Never miss a deadline again

LOG IN WITH IVLE

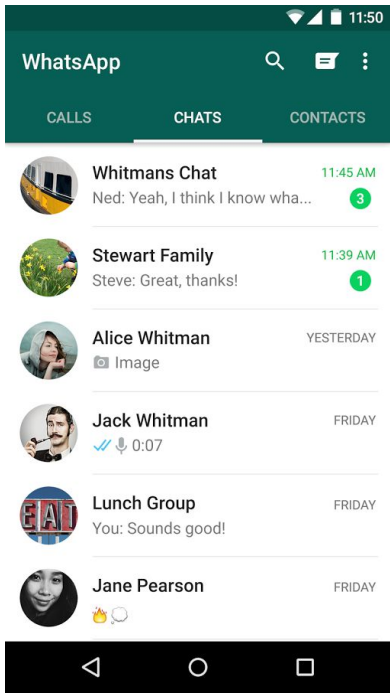
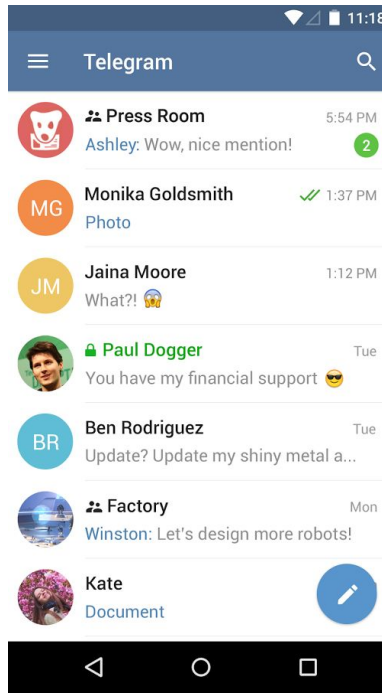



Splash page pc (width >= 1280px)

		
<p>Splash page (< 1280px)</p>	<p>Android (Chrome)</p>	<p>iOS (safari)</p>

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 the same feel when they use Schoolines. Different colors were used to represent different modules, and our website will support up to 10 different colors, which are all distinct. Since the maximum workload is 27mc for concurrent degree students, the 10 distinct colors should suffice , including special cases who are doing 7-9 modules in a semester.

		
WhatsApp	Telegram	Schoolines

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, most advice online are that a CSS methodology is overkill. However, we feel that at least loosely following a methodology, we have much more maintainable stylesheets, where similar objects can 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://drupaldruppings.com/sites/default/files/inline-images/google-chrome_your_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 the 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.

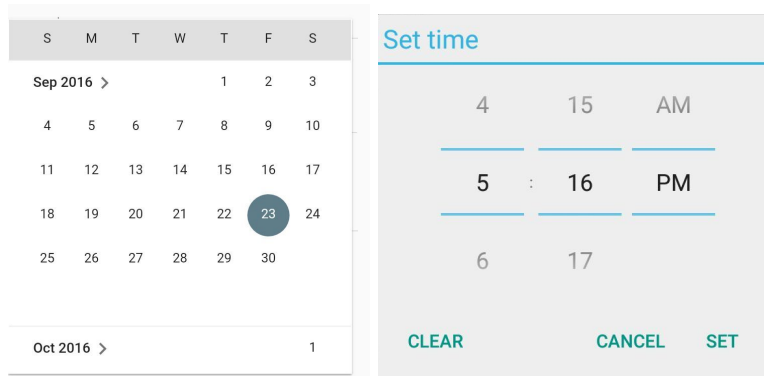
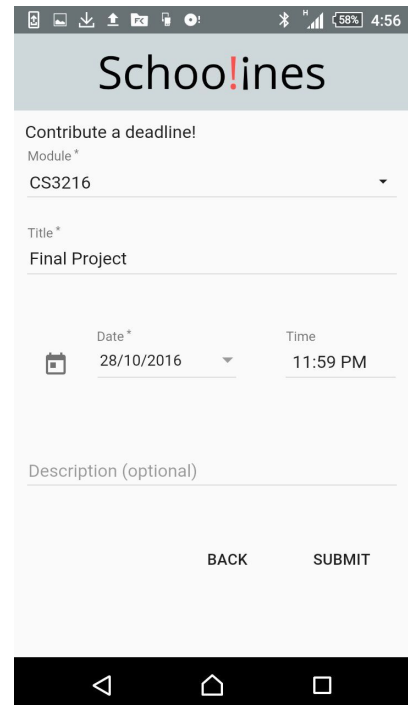
Milestone 12: Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. Lastly, list down some (at least 5) of the mobile site design principles and which pages/screens demonstrate them.

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.

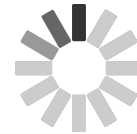
The image shows two side-by-side screenshots of web form components. The left screenshot is a date picker for September 2016, with the 23rd highlighted. The right screenshot is a 'Set time' dialog box showing the time 5:16 PM, with 'AM' and 'PM' options and 'CLEAR', 'CANCEL', and 'SET' buttons at the bottom.The image is a screenshot of a mobile application interface for 'Schoo!ines'. The title is 'Contribute a deadline!'. Below it is a 'Module *' dropdown menu with 'CS3216' selected. Then is a 'Title *' text input field with 'Final Project' entered. Below that are 'Date *' and 'Time' pickers, showing '28/10/2016' and '11:59 PM' respectively. There is a 'Description (optional)' text area. At the bottom are 'BACK' and 'SUBMIT' buttons. The status bar at the top shows various icons and the time 4:56.

(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.

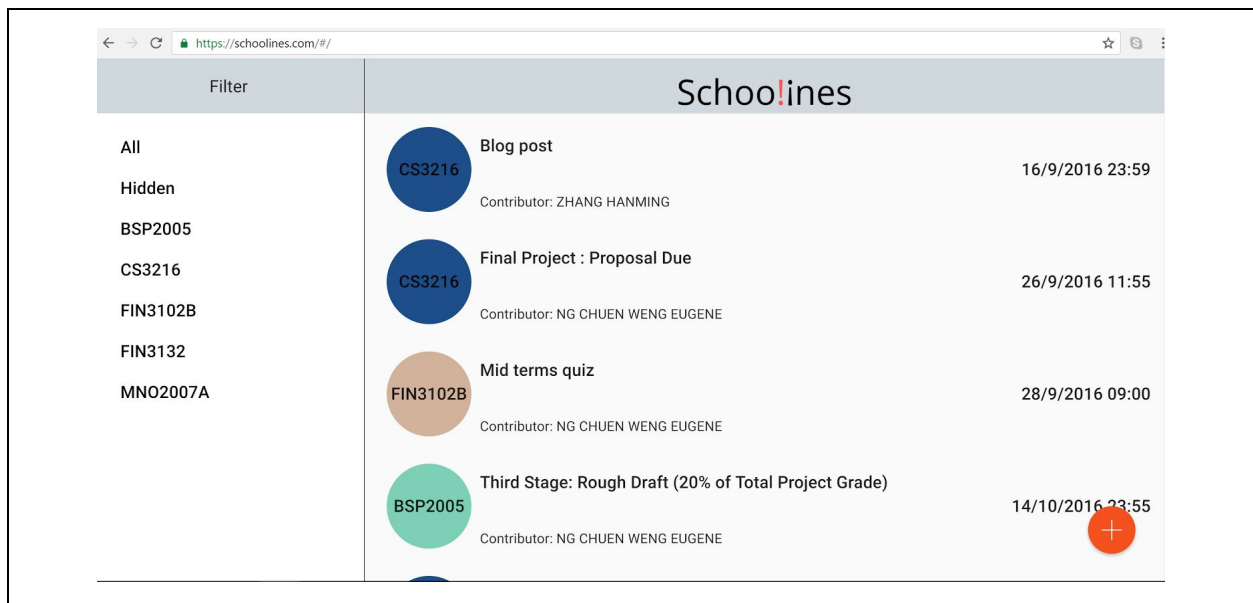


Milestone 13: Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

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.

User flow : Click on (+) button -> Fill up form -> Input Module -> Input Title -> Input Date -> Input time -> Input description -> Click submit



Contribute a deadline!

Module *

Title *

Date * 23/09/2016 Time * 23:59

Description (optional)

BACK SUBMIT

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.

Filter	Schoolines	
All	<div>CS3216</div> Blog post Contributor: ZHANG HANMING 16/9/2016 23:59	
Hidden		
BSP2005		
CS3216	<div>CS3216</div> Final Project : Proposal Due Contributor: NG CHUEN WENG EUGENE 26/9/2016 11:55	
FIN3102B		
FIN3132		
MNO2007A	<div>FIN3102B</div> Mid terms quiz Contributor: NG CHUEN WENG EUGENE 28/9/2016 09:00	
	<div>BSP2005</div> Third Stage: Rough Draft (20% of Total Project Grade) Contributor: NG CHUEN WENG EUGENE 14/10/2016 23:55	+

← → ↻

https://schoolines.com/#/deadlineDetail

☆

🔍

⋮

School!ines

CS3216

Blog post

By: ZHANG HANMING

Due on: 16/9/2016 23:59

BACK

HIDE

Filter

School!ines

All

Hidden

BSP2005

CS3216

FIN3102B

FIN3132

MNO2007A

CS3216

Final Project : Proposal Due

26/9/2016 11:55

Contributor: NG CHUEN WENG EUGENE

FIN3102B

Mid terms quiz

28/9/2016 09:00

Contributor: NG CHUEN WENG EUGENE

BSP2005

Third Stage: Rough Draft (20% of Total Project Grade)

14/10/2016 23:55

Contributor: NG CHUEN WENG EUGENE

CS3216

Final : Progress Report 1

17/10/2016 23:55

Contributor: NG CHUEN WENG EUGENE

+

Filter

School!ines

All

Hidden

BSP2005

CS3216

FIN3102B

FIN3132

MNO2007A

CS3216

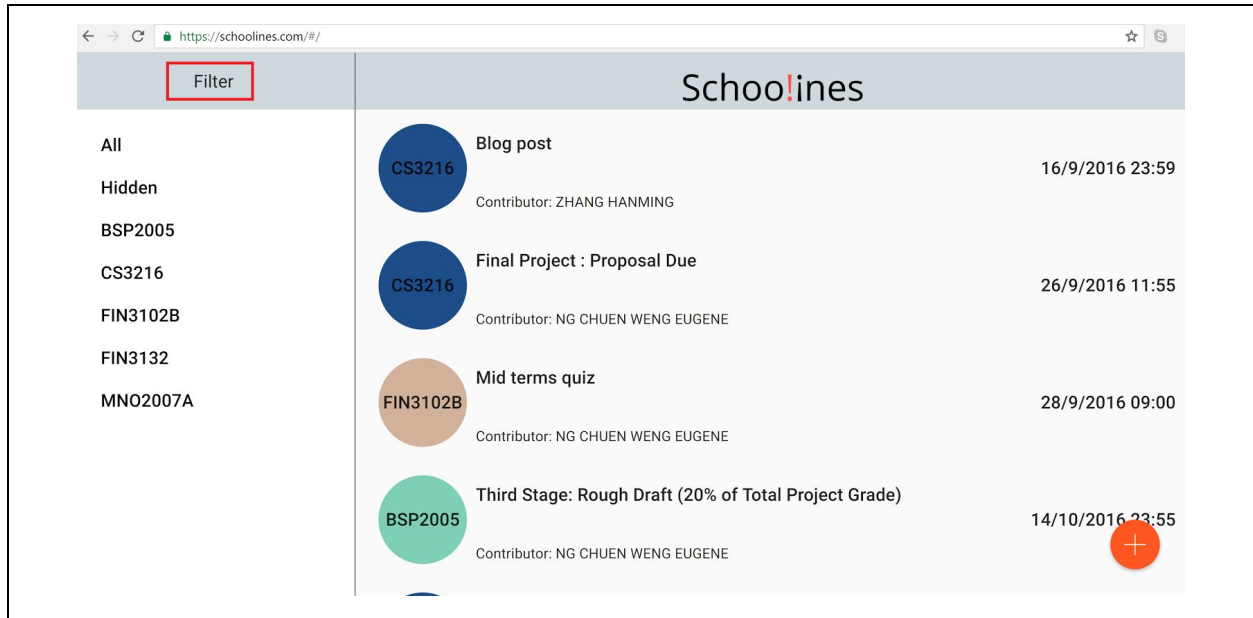
Blog post

16/9/2016 23:59

Contributor: ZHANG HANMING

+

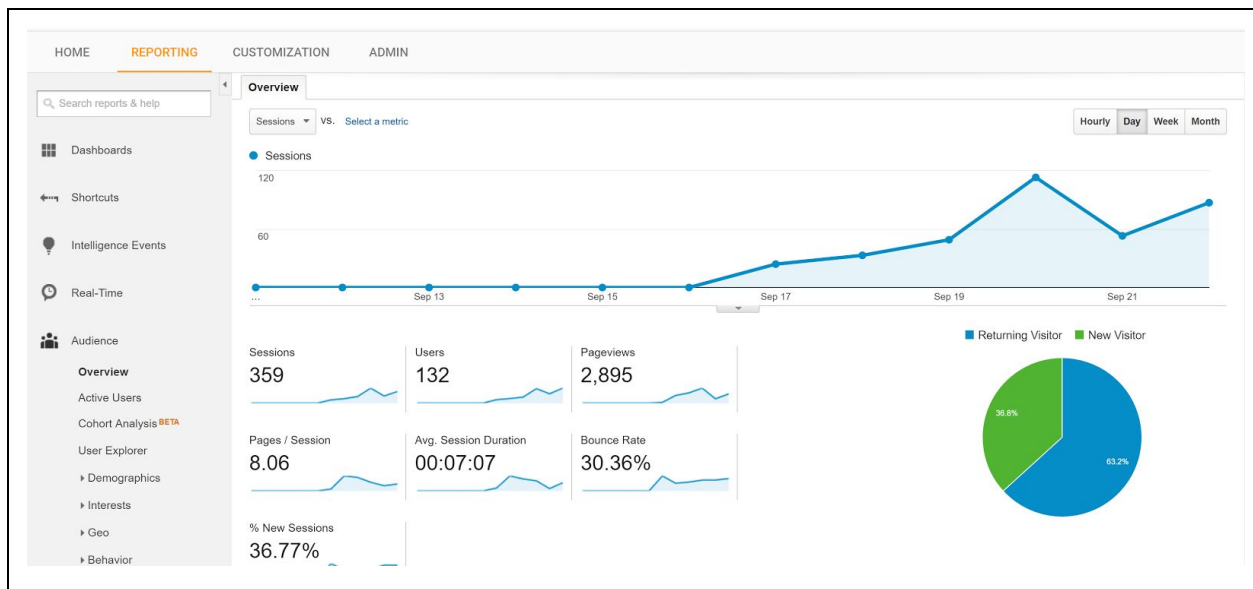
3. Filter deadlines : Allow users to filter deadlines based on modules. This helps users to identify deadlines pertaining to particular module and thus make it easier to keep track of deadlines.



The screenshot shows the Schoolines website interface. On the left, there is a sidebar with a 'Filter' button highlighted by a red box. Below the button, a list of modules is displayed: All, Hidden, BSP2005, CS3216, FIN3102B, FIN3132, and MNO2007A. The main content area, titled 'Schoolines', displays a list of deadlines. Each entry includes a colored circular icon with a module code, the task name, the contributor's name, and the deadline date and time. A red circular button with a white plus sign is located at the bottom right of the list.

Module	Task	Contributor	Deadline
CS3216	Blog post	ZHANG HANMING	16/9/2016 23:59
CS3216	Final Project : Proposal Due	NG CHUEN WENG EUGENE	26/9/2016 11:55
FIN3102B	Mid terms quiz	NG CHUEN WENG EUGENE	28/9/2016 09:00
BSP2005	Third Stage: Rough Draft (20% of Total Project Grade)	NG CHUEN WENG EUGENE	14/10/2016 23:55

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.

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