Infs3202 Proposal

Theory

Web Standards

Web standards exist to make websites cross-compatible, accessible, and easier to maintain. By adhering to web standards, your website/app automatically becomes cross-browser and cross-platform. As coding for different browsers and hardware can be incredibly time consuming, it is worthwhile to adhere to W3C standards from the get go.

Websites in accordance with web standards are accessible to everyone. The web should not discriminate against the disabled. Also, due to the nature of the W3C standards, any website made in accordance with them now will forever be accessible in the future, so as browsers evolve and change, the website will remain usable.

The Problem

One of the most common messages university students receive via Facebook is “are you free?”. Students who get out of lectures need someone to hang out with / eat lunch with / study with. To find that person, they generally message their best friends, asking the question “are you free?”, or “want to grab lunch?”. Most of the time, the reply is “I’m busy”, or “I’m not at uni”. These message exchanges don’t have to occur. What if you could find out who of your friends are free, or when they’re going to be free, at the press of a button?

SyncUQ

SyncUQ aims to be the first University of Queensland exclusive schedule sharing platform. Users will be able to share their timetables with friends and get live updates of when their friends are free. It also has the bonus functionality of allowing users to plan meetings with one another.

Technologies

In order to create this webapp, a stack of technologies was needed. To decide on this stack, many factors were considered. Among these factors were: Compatibility, Support, Ease of Use, and Cost.

The stack which best satisfied this was the following.

Frontend:

Purescript: Easy to use functional programming language that compiles to Html/JS

JavaScript: Needed for certain API's (such as Facebooks API)

Backend:

Python: A highly supported multi paradigm language, perfect for doing lots of things
Flask: A package for python created specifically for server side programming
SQLAlchemy: A package which allows python objects to function as SQL tables

Database:

Postgres: A very compatible DB with SQLAlchemy

Hosting:

Heroku: A PaaS that is extremely easy to use and scales with your app.

Design

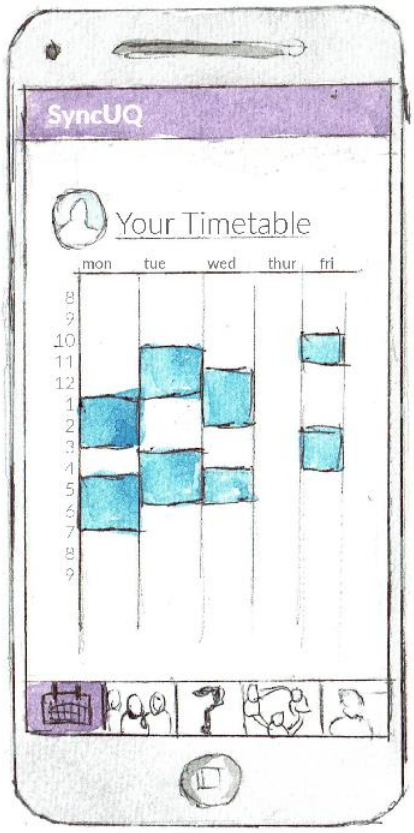
Colour Scheme

In order to represent the university, the primary colour choice for this webapp is the purple used by many UQ apps and services. A colour scheme was then built around this colour using a 'triad' approach.

Charlton Groves | 43960914 | UQ

Mobile Layout

This app is expected to be used by people 'on the go', and so a mobile first design is imperative.

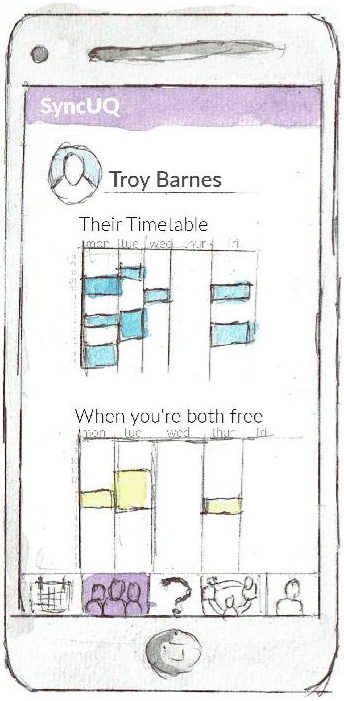


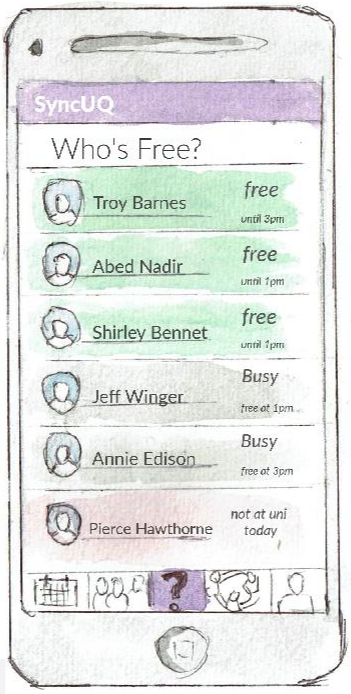
The UX/UI closely follows that of popular modern designs such as Spotify and Instagram. This approach utilises a simple icon based navbar at the bottom, with context specific buttons used to navigate from within the page.

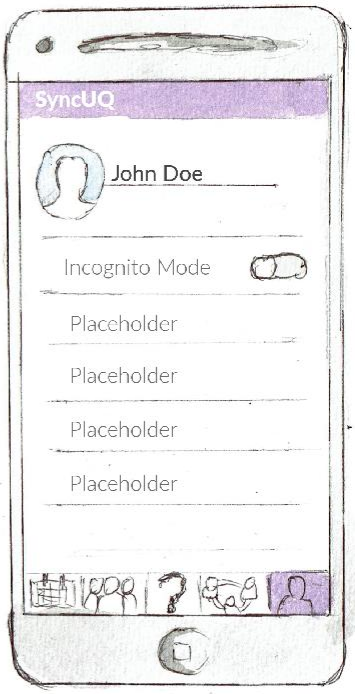
This design was chosen as the app at its core is relatively simple, and thus should have minimal navigational items.

The bottom nav bar navigates to the pages (from left to right):

- **Your Timetable** (view your timetable)
- **Friends** (view friends, add friends, accept friend requests)
- **Who's Free?** (A concise and informative list of who of your friends are free)
- **Groups** (Manage a group of friends' timetables)
- **Profile** (Manage your profile, as well as access security options)

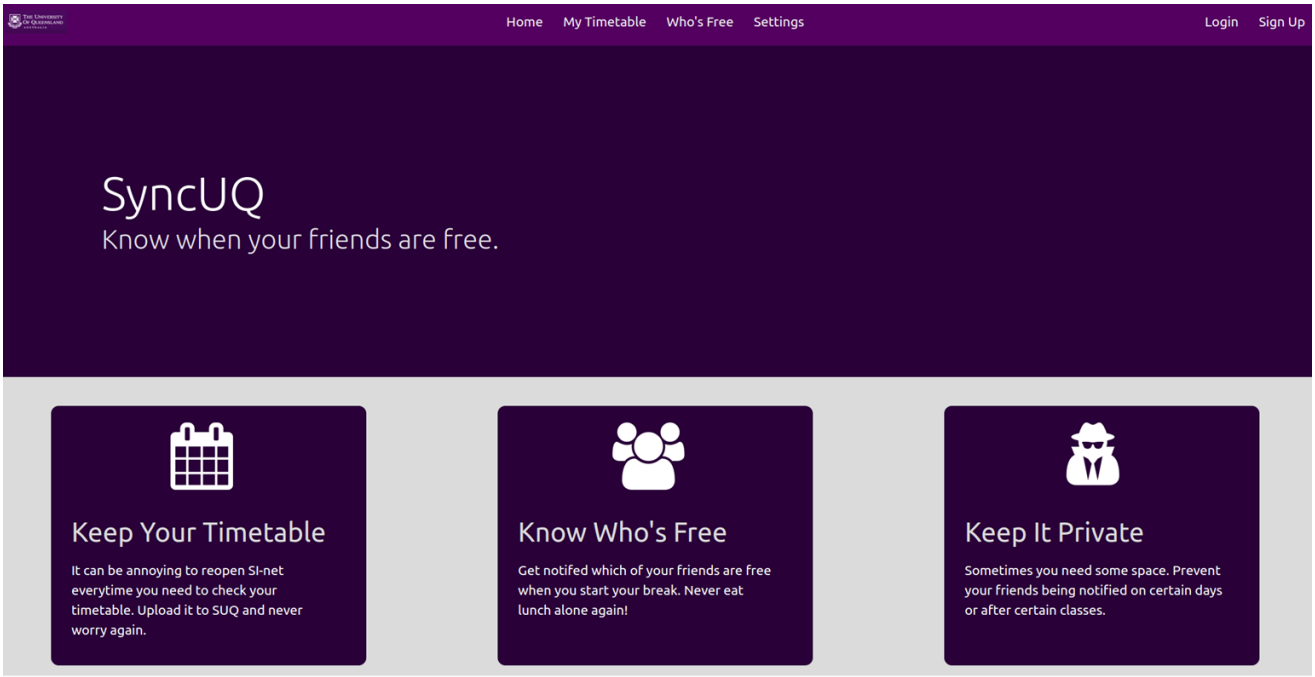






Website Layout

A functional website is still very important, as University students are often using a laptop or desktop computer. The website content and colour scheme is fundamentally the same as the mobile version. However, the layout has changed to a more desktop oriented design.



As you can see the nav has moved to the top. However, functionality remains the same.

Feature Coverage

Client Server Communication

User → server: User registers their account.

To do this, the user fills out a registry account and presses submit. The information is picked up via Flask and processed into the database via SQLAlchemy,

User → Facebook API → Server → User : User logs in via Facebook.

To do this, the user logs in via the Facebook button. The request is handled by a clientside front end script that asks Facebook for an access token and some user Info. All of this is then shipped in a JSON to the backend where it is unpacked by python and SQLAlchemy into the DB. This info is then served to the user when they view their profile

User → Sever → User : User uploads calendar and views it.

Calendars are uploaded either by a direct link to the UQ timetable planner timetable, or by uploading their file from their host machine. The calendar is then stored in the DB as a byte array. This info can then be sent back to frontend when the user views their calendar.

Charlton Groves | 43960914 | UQ

Server Side Code

User finds friends.

When a user clicks the ‘find Facebook friends’ button, a request for these friends is sent to the backend. First, the users Facebook ID is taken from their login session. This ID is then sent to Facebook’s graph API with a request for the user’s friends who are using SyncUQ. Facebook sends these Facebook ID’s back to the server. The server then runs a query against the registered user database, finding the user’s Facebook friends and displaying them in html.

User finds out whether their friends are free.

The user presses the “Who’s Free” tab. A query of all the user’s friends is run to gather a list of user id’s. One by one, each id has its calendar taken out of the database and processed. It is then determined whether each of the user’s friends are free, are going to be free, or are not at uni. These results are then served to the user via html.

Advanced JavaScript

Facebook login:

Using the Facebook api, a user can log in and have their info and access token sent back to them. The javascript function then puts all this information into JSON to send back to the server.

Calendar view:

The server passes a JSON ‘calendar’ to the frontend. A complex javascript script then processes this into something the user can view and easy comprehend.

Animations:

JQuery will be used to create animations for toggles, page change transitions, and context specific icons.