# BCCH Internal Design

## Team Info

**Team Name**: Future Star
**Team Members**: Nick Birtch, Lang Cheng, Jiayao Chen, Kaiti Mok Rong, Alexander Svatukhin, Jiawei Tan, Julie Zhu

## Document Information

**Revision History**

| Date | Version | Status | Prepared by | Comments |
|---|---|---|---|---|
| Feb 5th, 2020 | 0.0.1 | Initial outline/notes. | Jiayao Chen | N/A |
| Feb 6th, 2020 | 0.2.0 | Finished Programming environment | Jiayao Chen | N/A |
| Feb 6th, 2020 | 0.3.5 | Introduction, notable risks | Jiawei Tan, Nick Birtch | N/A |
| Feb 6th, 2020 | 0.6.0 | Added API design. Half-way done Production & testing environment, software architecture | Lang Cheng, Jiayao Chen, Kaiti Mok Rong | N/A |
| Feb 7th, 2020 | 0.7.0 | ER diagram & data table design | Lang Cheng, Jiayao Chen | N/A |
| Feb 7th, 2020 | 0.8.5 | UI design & mock up | Jiayao Chen | N/A |
| Feb 7th, 2020 | 1.0.0 | Noticeable trade offs, production & testing environment | Lang Cheng, Kaiti Mok Rong | N/A |
| Feb 23rd, 2020 | 1.1.0 | Feedback Revision | Nick Birtch, Kaiti Mok Rong, Jiayao Chen | Minor readability and content updates, Rewrite Database Design/ER Diagram, Added algorithm section. Class diagram |
| Apr 6th, 2020 | 1.2.0 | API endpoints / DB table updates | Jiayao Chen | N/A |

**Document Control**

| Role | Name | E-mail | Telephone |
|---|---|---|---|
| Project Manager | Jiayao Chen | raymondyao1997@gmail.com | 778-858-2077 |

## Introduction

The H-Behaviors Lab in BC Children's Hospital would like to develop a new way to collect patients' assessments by a combination of new video technologies as well as the Internet. This enables patients' observations in a home setting and greatly reduces the difficulties of disorder diagnosis and continuous monitoring of treatment regiments.

This project aims to build an easy-to-use web application for patients to report their health condition as well as for medical staff (admin) to keep track of patients (user). The web application will include a login page, a dashboard page and an upload page. Login page will only include the basic logic for patients and doctors to log in, and authentication logic will be handled by BCCH. As for the dashboard, it will allow patients to see their own data and it will allow doctors to access, manage and export all patients' information and reports by querying (such as filtering and grouping). Last but not least, the upload page will allow patients to upload assessments (selfies, videos, surveys) while supporting the additional admin functionality of customizing assessment templates.

**Goals Overview**

- To achieve a minimum viable product, our team will first develop a mobile friendly web-based data collection system for the WeSleepSmart Research Project, which supports the storage, retrieval and analysis of patients' assessments, including sitting, playing/gaming and sleeping data.
- To provide better user experience on mobile, our team will extend the project to have native mobile support, using either a cross-platform technology such as React Native or native development toolkits for iOS and Android.

**Assumptions**

- It is assumed that the project will be mobile friendly and web based with a possible extension to native support. No guarantee to support native iOS or Android applications.
- It is assumed that the security of patient login credentials will be handled separately by BCCH.
- It is assumed that videos will be using MPEG-4 covert (mp4 format) and photos will be using JPEG format.

## Programming Environment

Programming Language:
This project will be a typical Node.js project whose frontend and backend will be developed using dialects of JavaScript. With the help of existing open source compilers, both dialects will be compiled back to JavaScript satisfying ECMAScript 2015 (ES6) standard, which is supposed to be fully supported by Node.js after version 8.0 and Google Chrome after version 58:

- Back End: The server side of this project will be developed in TypeScript 3.7.5
- Front End: The client side of this project will be developed in JSX, an XML-like syntax extension to JavaScript dedicated to React

Main Frameworks & Libraries:

- HTTP framework: Express.js, with some of its middlewares such as express-session management.
- Client framework: React v16
- Client Style: SCSS
- Client Ajax library: axios.js
- Unit Testing: Mocha
- Performance testing/UI automated testing:  Selenium
- Browser Support: Chrome (after version 58), Safari

Build Tools:

- Back End: TypeScript default compiler, tsc
- Front End: Webpack with Babel loader and SaSS loader

Database:
This project will use MySQL version 5.7

IDE, source control system & other tools:
The main development IDE/Editor will be Visual Studio Code. The project is a typical Git project and tracked on:

- GitLab (main repository): https://gitlab.com/cpsc319-2019w2/bc-childrens-hospital/futurestars/319-bcch
- GitHub (Mirror repository for backup purpose): https://github.com/Raymond-yao/319-BCCH

For the purpose of development and API testing, Postman will be used to initiate RESTful requests.

## Production & Test Environment

---

Production Environment

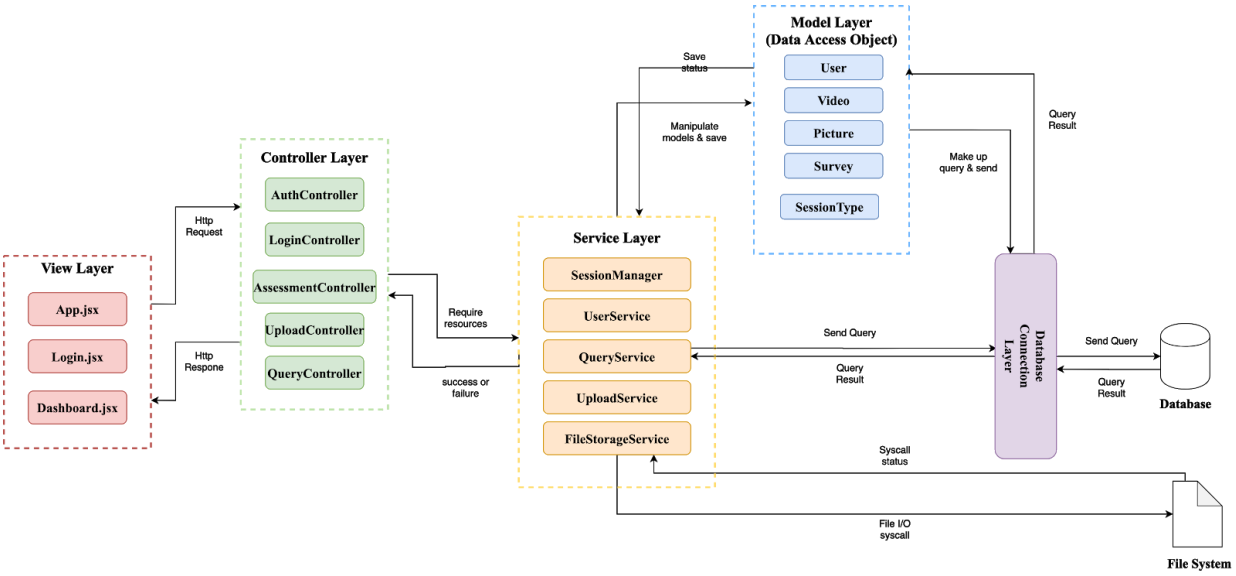
A seperate online server (Amazon Web Service etc.) will be used to run the application. During this phase of the project, the application should be completed and runnable. During this phase we will mostly conduct tests on the security and UI of the program. Further stress-testing will also be performed in the production environment.

Test Environment
Tests will be conducted on our local server. Separate mock data will be prepared to test the frontend, backend, and database for our application. We will test the overall functionality of the application and tests runned should mimic the actual setting where the application would run during runtime. Tests will be runned on both ios and android systems. Hardware devices such as the computers will be provided by our own team. Debugging and upgrade will also be done in our test environment.

## Software Architecture

---

1. This project will be a typical RESTful web application which has a MVC (Model, View & Controllers) structure.
2. The view layer initiates a HTTP request to the server, which will be first handled by the corresponding controller and processed by suitable services.
3. Each service will mostly interact with data access objects, an object representation of database entity to avoid direct SQL manipulation.
4. There are two exceptions, however, which will have to deal with raw SQL query and file systems due to special requirements. In particular, QueryService might need to have access to full SQL control in order to provide flexibility for the researchers and FileStorageService has to keep in touch with file systems in order to store large media files.
5. An extra Database connection layer is added to abstract out the implementation details of connecting to MySQL, sending queries and receiving data.




## Data Design

---

Table design:

- User(<u>id</u>, name, display_name, password, age, gender, sex, date_created, is_admin, date_of_birth)
- Video(<u>video_id</u>, **assessment_id**, **user_id**, filename, size, time_created, is_archived)
- VideoDescription(<u>video_desc_id</u>, **assessment_temp_id**, desc)
- Picture(<u>pic_id</u>, **assessment_id**, **user_id**, filename, size, time_created, is_archived)
- PictureDescription(<u>pic_desc_id</u>, **assessment_temp_id**, desc)
- SurveyTemplate(<u>sur_temp_id</u>, sur_temp_name, instruction, time_created)
- Survey(<u>sur_id</u>, **assessment_id**, **user_id**, **sur_temp_id,** time_created, is_archived)
- SurveyQuestion(**<u>sur_temp_id</u>**, <u>question_num</u>, question_type, statement, meta)
- SurveyAnswer(**<u>sur_id</u>**, <u>question_num</u>, **sur_temp_id**, user_id, answer)
- HasSurvey(**<u>assessment_temp_id, sur_temp_id</u>**)
- Assessment(<u>assessment_id</u>, **session_id**, **assessment_temp_id,** is_archived)
- AssessmentTemplate(<u>assessment_temp_id</u>, assessment_temp_name, description, num_videos, num_pics, num_surveys, time_created, is_archived)

*<u>underlined</u> fields are primary keys

***bolded** fields are foreign keys

## API Design

---

- Login page: **GET** /
    - Result
        - Success **200**:HTML
        - Not Found **404**: Error

- Authentication: **POST** /login
    - Body: {username, password}
    - Result
        - Success **200**: JSON {user_id, cookies}
        - Unauthorized **401**: Invalid username or password
- Login page: **GET** /userInfo
    - Result
        - Success **200**: JSON
        - Unauthorized **401**: Invalid username or password
- Upload File: **POST** /logout
    - Body: None
    - Result
        - Success **200**: OK
- Get Session: **GET** /assessment/all
    - Description: Get assessments pre-defined by researchers
    - Result
        - Success **200**: JSON {list of predefined surveys}
        - Not Found **404**: Result: Error
- Get Session: **GET** /assessment/{id}
    - Result
        - Success **200**: JSON {session info}
        - Not Found **404**: Result: Error
- Create Session: **POST** /assessment/add
    - Description: Create assessment
    - Body: {[assessment_id], int: number of videos/selfies}
    - Result
        - Success **200**: assessment_id
        - Not Found **404**: Result: Error
- Delete Session: **DELETE** /assessment/{id}
    - Description: Delete assessment
    - Params
        - id: assessment_id
    - Result
        - Success **200**: OK
        - Not Found **404**: Result: Error
- Get Sessions: **GET** /surveys/all
    - Description: Get surveys pre-defined by researchers
    - Result
        - Success **200**: JSON {list of predefined surveys}
        - Not Found **404**: Result: Error
- Get Survey: **GET** /survey/{id}
    - Params

        - id: survey_id (sur_id)
    - Result
        - Success **200**: survey_file
        - Not Found **404**: Error
- Create Survey: **POST** /survey/add
    - Body: {survey entries, survey type}
    - Result
        - Success **200**: survey_id
        - Not Found **404**: Error
- Query data: **POST** /upload/start/{id}
    - Params:
        - id: assessment_template_id
    - Result:
        - Success **200**: JSON {id: assessment_id}
        - Unauthorized **401**: Invalid username or password
- Query data: **POST** /upload/video/{id}
    - Params:
        - id: assessment_id
    - Body: Submit form with video streaming data
    - Result:
        - Success **200**: OK
        - Unauthorized **401**: Invalid username or password
- Query data: **POST** /upload/picture/{id}
    - Params:
        - id: assessment_id
    - Body: Submit form with picture streaming data
    - Result:
        - Success **200**: OK
        - Unauthorized **401**: Invalid username or password
- Query data: **POST** /upload/survey/{id}
    - Params:
        - id: assessment_id
    - Body: {survey_answer}
    - Result:
        - Success **200**: OK
        - Unauthorized **401**: Invalid username or password
- Query data: **POST** /query/media
    - Body: {query}
    - Result:
        - Success **200**: JSON {data after query}
        - Bad Request **400**: Error (invalid query)
- Query data: **POST** /query/survey

        - Body: {query}
        - Result:
            - Success **200**: JSON {data after query}
            - Bad Request **400**: Error (invalid query)
- Query data: **POST** /query/plain
    - Body: {query}
    - Result:
        - Success **200**: JSON {data after query}
        - Bad Request **400**: Error (invalid query)

## Algorithms

---

As we are supporting a typical database driven application, there is minimal opportunity for complex algorithms in the scope of our project. We will make an effort to extend and improve on the performance of our queries with our database design choices while taking advantage of database indexes and query algorithm techniques.

## Notable Trade offs

---

CPU Time

- Converting video to MP4 format will increase CPU time needed
- Converting picture from a different format will increase CPU time needed

Quality

- Converting video/picture may cause lower resolution and quality

Storage

- Since storage is provided by stakeholders (BCCH), we assume there is no trade off for this part.

Security

- Access for videos/pictures will be handled separately, those APIs may be vulnerable under certain circumstances, such as improper authentication.

## Notable Risks

---

The primary risks involved in not meeting the system requirements are:

Performance degradation under load:

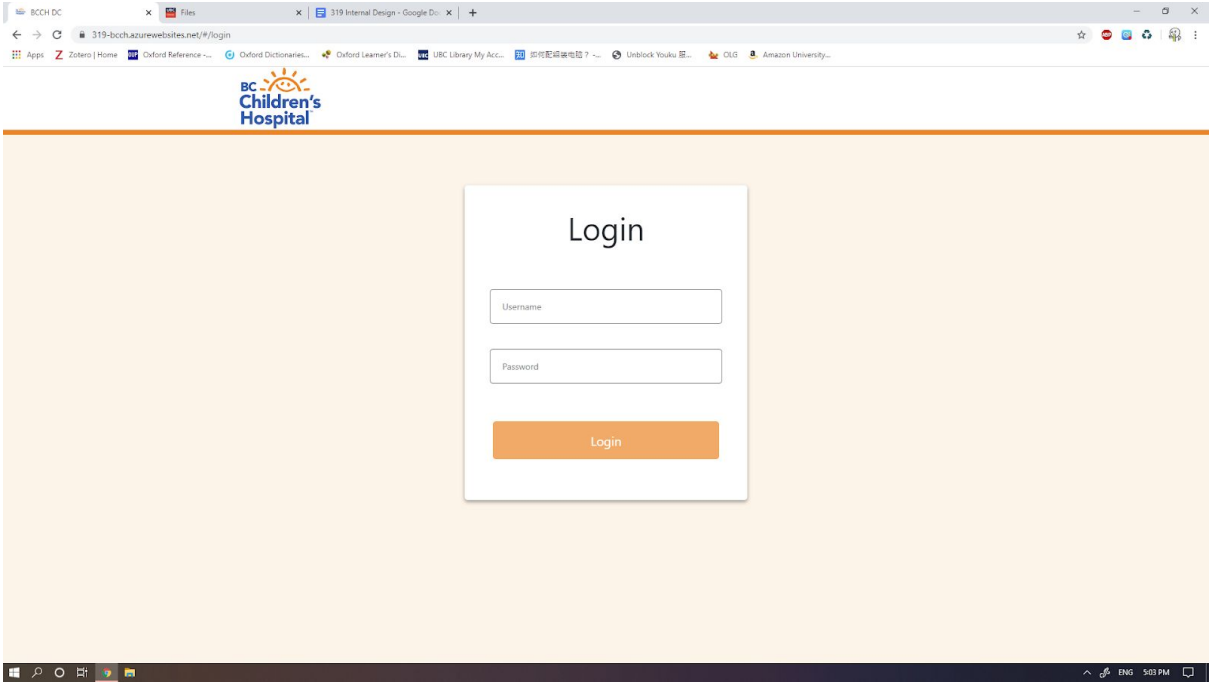
- Overall performance must be reasonably quick and be robust enough to handle variant internet speeds, multiple connections and the perpetually growing database.
- To combat these concerns, we will prioritize improvements and optimizations of the implementation of our database to file system connection, data queries and API throughout development.

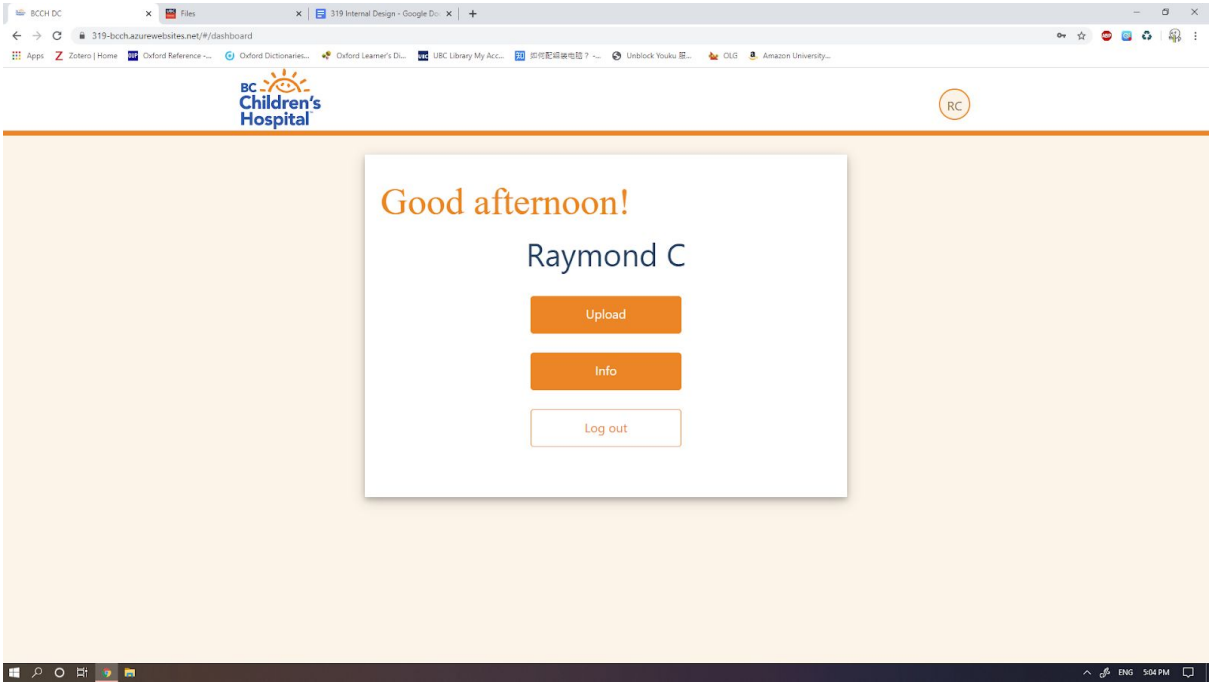
- As the performance benchmarks are well defined, we will include a heavy emphasis on performance in the QA testing suite while monitoring and addressing any significant issues.

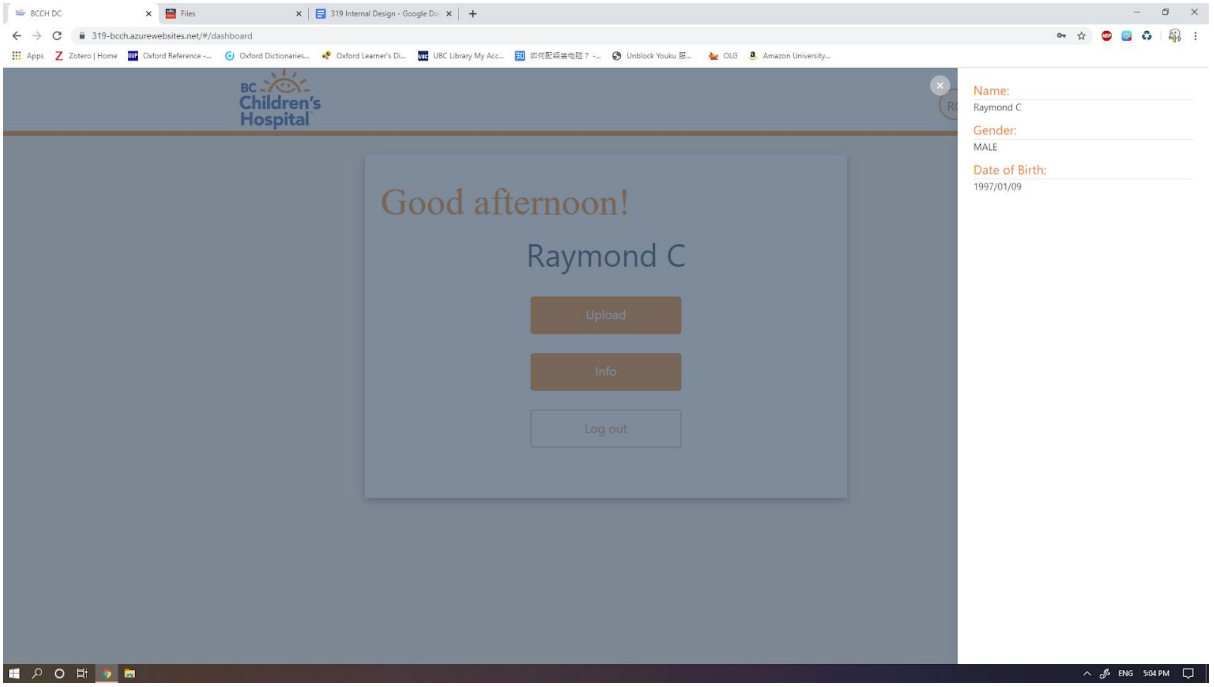
Poor mobile performance and usability:

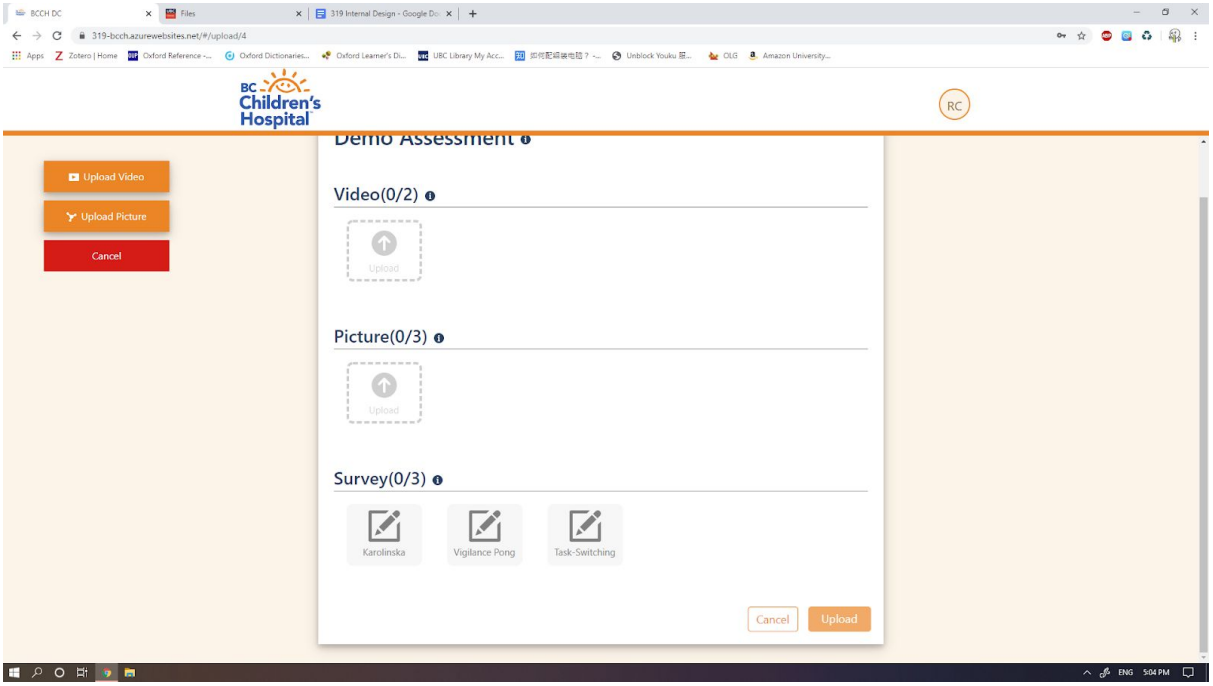
- BCCH places high value on mobile functionality, which ties in with the photo/video based aspect of the assessments for ease of use.
- We have introduced a Mobile Compatibility Product Owner to ensure that the front end design is reliable and mobile friendly and that acceptance testing using a mobile device is a consistent part of our testing process.
- We will focus on the minimum viable product of a mobile friendly web-based data collection system. If time and project stability permits, we will attempt a mobile specific prototype.

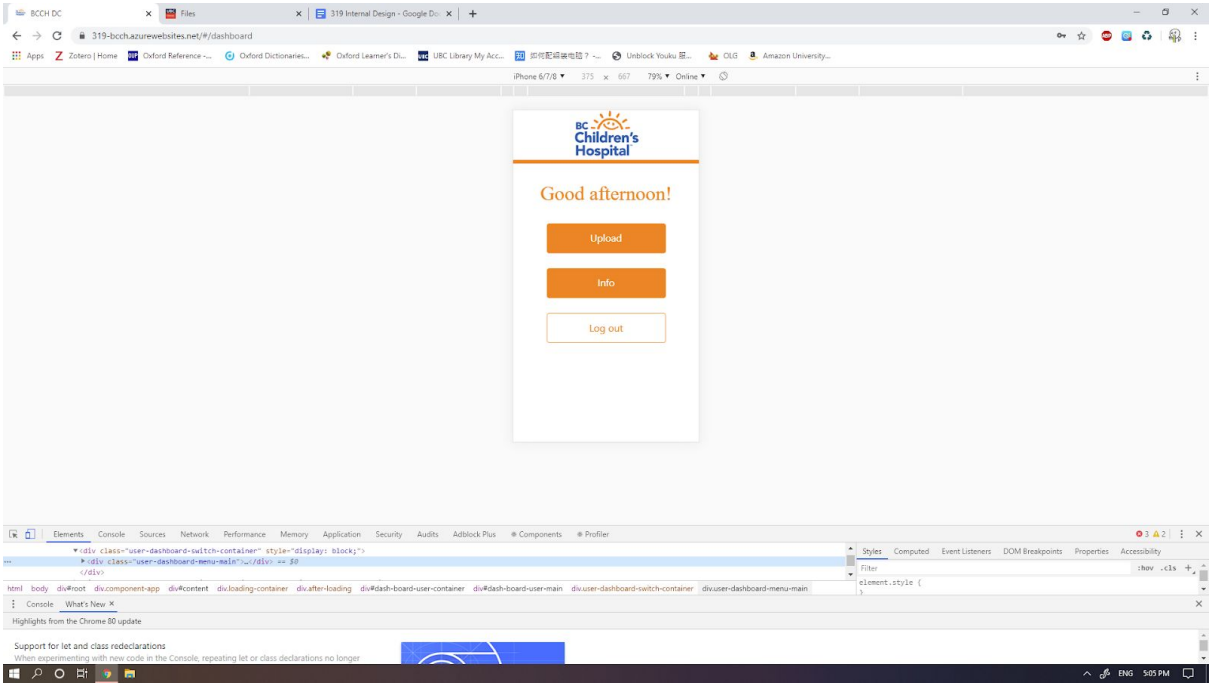
## UI Design & Mock up

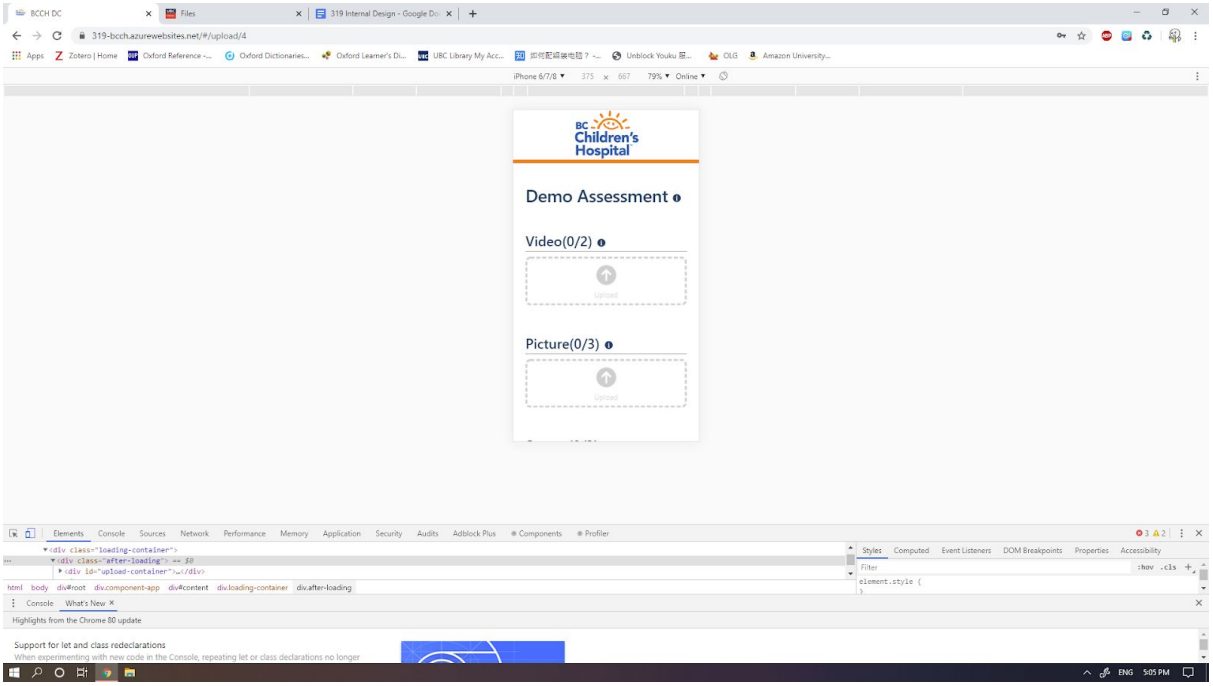



BC Children's Hospital
RC
Good afternoon!
Raymond C
Upload
Info
Log out


BC Children's Hospital
Good afternoon!
Raymond C
Upload
Info
Log out
Name:
Raymond C
Gender:
MALE
Date of Birth:
1997/01/09


BC Children's Hospital
RC
Upload Video
Upload Picture
Cancel
Demo Assessment
Video(0/2)
Upload
Picture(0/3)
Upload
Survey(0/3)
Karolinska
Vigilance Pong
Task-Switching
Cancel
Upload


BC Children's Hospital
Good afternoon!
Upload
Info
Log out



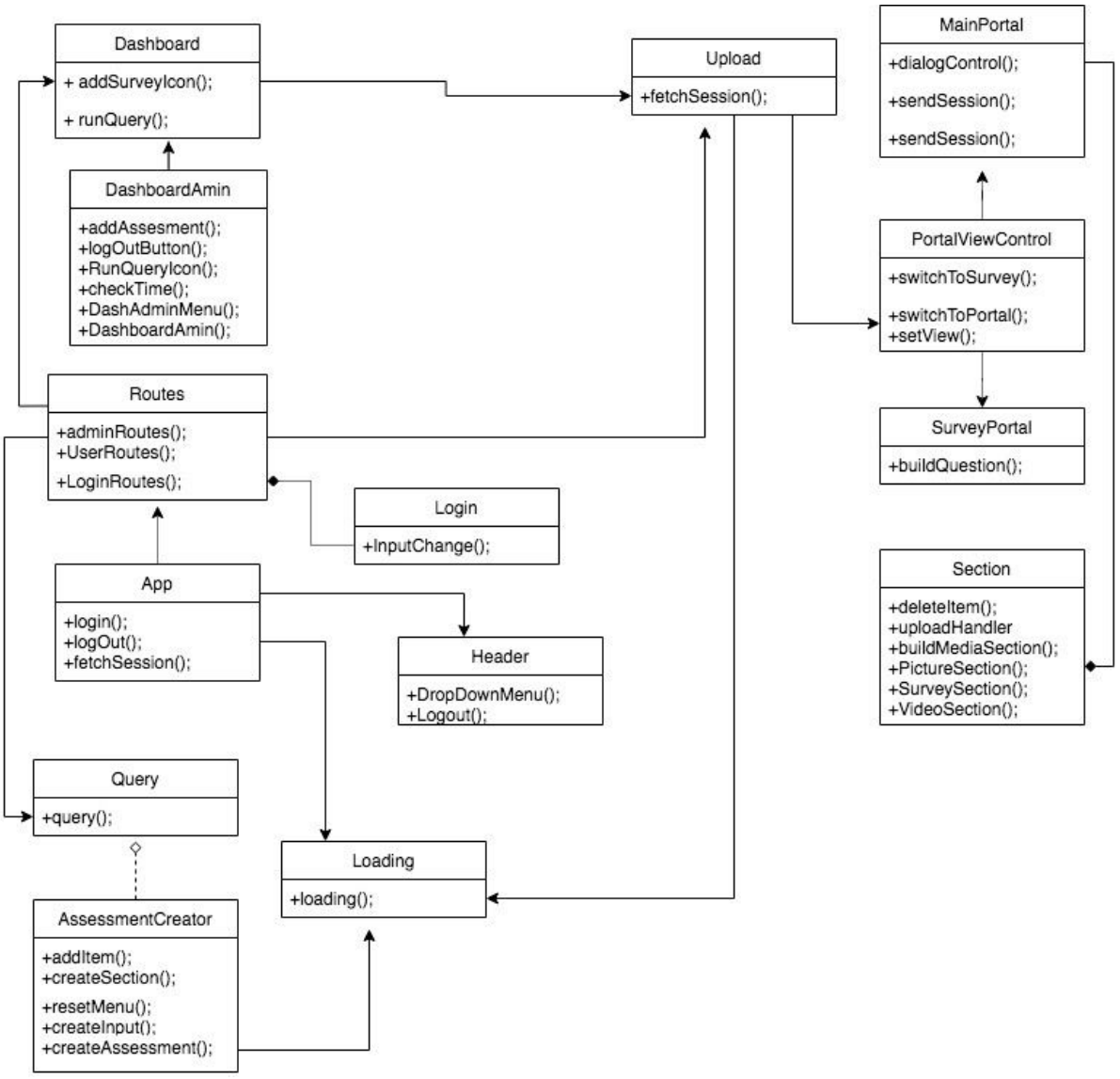

## Support Specs

---

Change Request Forum : https://drive.google.com/file/d/1mik_O5OwGXw-9TvYw4HpPJ8D9rISGXQ_/view?usp=sharing

## Class diagram

---


Dashboard
+ addSurveyIcon();
+ runQuery();
DashboardAmin
+addAssesment();
+logOutButton();
+RunQueryIcon();
+checkTime();
+DashAdminMenu();
+DashboardAmin();
Upload
+fetchSession();
MainPortal
+dialogControl();
+sendSession();
+sendSession();
PortalViewControl
+switchToSurvey();
+switchToPortal();
+setView();
SurveyPortal
+buildQuestion();
Routes
+adminRoutes();
+UserRoutes();
+LoginRoutes();
Login
+InputChange();
App
+login();
+logOut();
+fetchSession();
Header
+DropDownMenu();
+Logout();
Section
+deleteItem();
+uploadHandler
+buildMediaSection();
+PictureSection();
+SurveySection();
+VideoSection();
Query
+query();
Loading
+loading();
AssessmentCreator
+addItem();
+createSection();
+resetMenu();
+createInput();
+createAssessment();