Turbo Consulting

Turbo Consent

Portfolio B

April 2018

Contents

1 Project Overview 2
2 System Requirements 3
2.1 System Stakeholders 3
2.2 Functional Requirements 3
2.3 Non-Functional Requirements 4
2.4 Use Case Diagram 4
2.5 Key Use-Cases 5
2.6 Core Use-Case Flowchart 7
3 System Architecture 8
4 Testing Strategy 9
4.1 Development Testing 9
4.2 Release Testing 11
4.3 Acceptance Testing 12
5 Object-Oriented Design & UML 13
6. User Interface Design 15

Tables

Table 1: Functional Requirements 3
Table 2: Non-functional Requirements 4
Table 3: Key Use-Cases 6
Table 4: Development Testing Test Cases 10
Table 5: Release Testing Test Case 12

Figures

Figure 1: Use Case Diagram 4
Figure 2: Core Use-Case Flowchart 7
Figure 3: High-Level System Architecture Diagram 8
Figure 4: Layered Architecture UML Class Diagram 13
Figure 5: Entity Relationship Diagram 13
Figure 6: Business Layer UML Class Diagram 14
Figure 7: Initial Webpage Design Wireframes 15
Figure 8: Current Experiments Page 15
Figure 9: Current Consent Selection Page 16
Figure 10: Mobile View of Homepage and Visitors Page 16

1 Project Overview

Turbo Consulting are working alongside We The Curious on the development of Turbo Consent, a web application that will allow visitors of We The Curious to understand and manage their data consent options in a quick, easy and clear manner once they have taken part in live research at the Science Centre.

The Client
We The Curious Science Centre (formerly At-Bristol) welcomes over 300,000 visitors every year to explore science, arts, culture, industry and the community through the exciting exhibitions and activities that they host at the centre. For 17 years, their aim has been "making science accessible to all" and now with a new direction and focus, they aim to "create a culture of curiosity".

The Problem
As part of the new developments at We The Curious, in 2020, a new area will be opened called Open City Lab, where visitors of the centre will be able to partake in real research and experiments conducted by researchers on the floor.

For researchers to use the data that they collect in their experiments, visitors must give their consent. This is a significant challenge as some of the data collected by researchers may potentially be sensitive - for example, health data or genetic data. Therefore, it is very important that We The Curious handles visitors' data correctly and adheres to the highest ethical standards. However, We The Curious wants anyone to be able to participate in experiments in Open City Lab without having to sign a consent form. It is also important that parents have the ability to give consent for their children even if their children come as part of a school group without their parents being present.

The Solution
Turbo Consulting will be developing Turbo Consent, a web application that will empower Open City Lab participants to understand and manage their consent to share their data. This web app will provide users with clear information about how their data will be used and what consent options are available to them. Users will have the ability to select their preferred consent options for each participated-in experiment at Open City Lab, for all the visitors under their control (themselves and/or their children). If research participants are too young to give consent for their data to be used, their parent or guardian will be able to use Turbo Consent on behalf of their child. Users will be able to view their previous choices, and set consent defaults if they do not want to provide explicit consent options each a visit is made to Open City Lab.

Turbo Consent will have a clean, simple interface and be easy to use for everyone. The web application will be completed in April 2018.

2 System Requirements

2.1 System Stakeholders

- **We The Curious (WTC):** The client of Turbo Consulting. Hosts the live research experiments. Owns and manages the Turbo Consent web application.
- **Turbo Consulting:** Developers of the Turbo Consent web application. Sets up and provides detailed system documentation of the web application before handing over the application to WTC.
- **WTC Visitors Under Consent Age:** Ineligible as a Turbo Consent account holder. Required to sign up as a dependent of a Turbo Consent account holder before taking part in live experiments in the Open City Lab.
- **WTC Visitors Above Consent Age:** Eligible as a Turbo Consent account holder. Take part in live experiments hosted by WTC and choose consent levels for the experiments results. Can give data consent on behalf of their dependent under-aged visitors.
- **Researchers:** Partners of WTC which conduct the experiments in the science centre for public participation. Only allowed to use experiment results once consent has been given on Turbo Consent.

2.2 Functional Requirements

Requirement ID	Requirement Description
FR1	The system shall allow users to login using an email and password.
FR1	The system shall allow admin users to create and delete accounts.
FR2	The system shall allow admin users to create Visitors and add them to an account.
FR3	The system shall allow admin users to create and delete experiments with experiment details and custom consent options.
FR4	The system shall allow admin users to add an experiment to a visitor once an experiment is participated-in at the Open City Lab.
FR5	The system shall allow users to view experiment information and select a data consent option for each participated-in experiment, for each visitor under the users' control.
FR6	The system shall allow users to change the default consent option for each visitor under the users' control.
FR7	The system shall allow users to view and change the consent options for all previously reviewed experiments, for each visitor under the users' control.
FR8	The system shall allow users to give a bulk consent option for multiple experiments at the same time.
FR9	The system shall allow users to logout.

Table 1: Functional Requirements

2.3 Non-Functional Requirements

Requirement ID	Requirement Description
NFR1	The system should respond to a click within 10 seconds.
NFR2	The system should allow up to 500 concurrent users.
NFR3	The system should allow up to 1,000,000 consent option selections to be stored in a database.
NFR4	The system shall allow users to only access the data associated with their account. The system shall deny access to any user attempting to access or submit any data not associated to their account.
NFR5	The system shall only allow users with admin authority to create or delete accounts, visitors and experiments. The system shall deny access to any user without admin authority attempting to perform these actions.

Table 2: Non-functional Requirements

2.4 Use Case Diagram

The following figure depicts the key actors who engage with the Turbo Consent system and the primary functional goals they wish to achieve. The sequence of steps needed to achieve each of the use-cases shown in this figure are listed in Table 3.

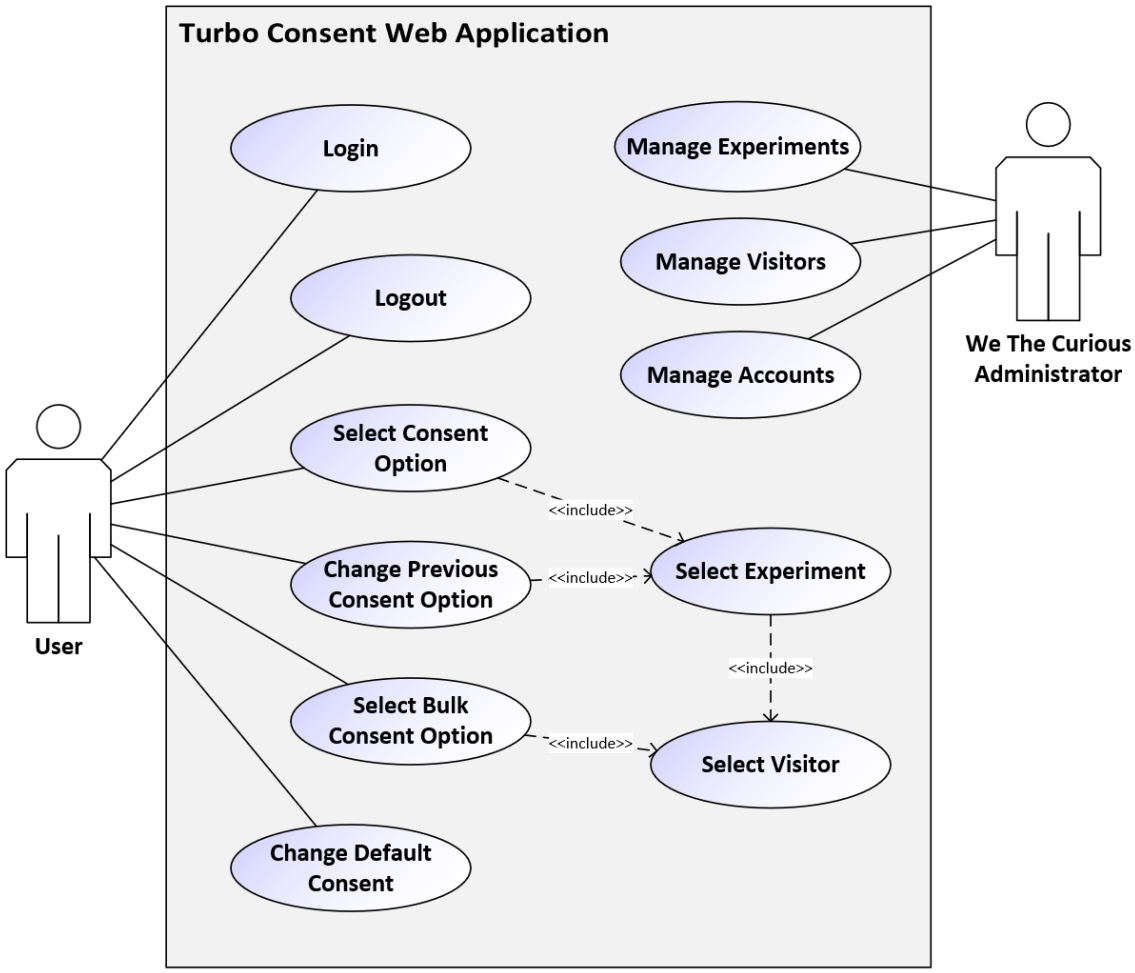

Figure 1: Use Case Diagram

2.5 Key Use-Cases

The following table shows the key use-cases of the Turbo Consent web application and the sequence of steps involved in achieving each goal.

Use-Case	Description	Steps
UC1	Login	1. Unauthenticated user tries to access the Turbo Consent homepage. The user is redirected to the login page. 2. User fills in log-in credentials and clicks 'Login'. The user is redirected to the web page they attempted to access. **Exceptional Flow:** If the inputted login credentials are invalid, display error message.
UC2	Select Consent Option	1. User logs in to the Turbo Consent web application and is directed to the homepage. 2. User clicks "My Visitors". 3. User selects a visitor from the displayed list. 4. User selects an experiment from the displayed list. 5. User is shown information about the selected experiment. The user selects a consent option and clicks submit. Their response is saved to the database.
UC3	Change Previous Consent Option	1. User logs in to the Turbo Consent web application and is directed to the homepage. 2. User clicks "My Visitors". 3. User selects a visitor from the displayed list. 4. User clicks on the "Reviewed Experiments" tab. User is shown a list of experiments along with the consent option that was chosen for each. 5. User selects an experiment from the displayed list. 6. User selects a new consent option and clicks submit. Their new response is saved to the database.
UC4	Select Bulk Consent Option	1. User logs in to the Turbo Consent web application and is directed to the homepage. 2. User clicks "My Visitors". 3. User selects a visitor from the displayed list. The user is shown a list of experiments that the selected visitor has participated in. 4. The user clicks "Select All" or the checkbox next to the experiments they want to give bulk consent for. 5. The user selects a consent option and clicks "Apply Consent Option". Their response is saved to the database.
UC5	Change Default Consent	1. User logs in to the Turbo Consent web application and is directed to the homepage. 2. User clicks "Settings". 3. User clicks "Select All" or clicks the checkbox next to the visitors they wish to change the default consent option for. 4. User selects a consent option and click "Apply Consent Option". Their response is saved to the database.

Use-Case	Description	Steps
UC6	Manage Accounts	1. Administrator logs in to the Turbo Consent web application and is directed to the homepage. Administrator navigates to the admin page. 2. Administrator clicks “Manage Accounts”. 3. Administrator deletes existing accounts or creates new account by filling in account details and clicking “Create”. The changes are saved to the database. **Exceptional Flow:** If the administrator attempts to delete their own account, prevent action and display error message.
UC7	Manage Visitors	1. Administrator logs in to the Turbo Consent web application and is directed to the homepage. Administrator navigates to the admin page. 2. Administrator clicks “Manage Visitors”. 3. Administrator deletes existing visitors or creates new visitor and assigns it to an account by filling in visitor details and clicking “Create”. The changes are saved to the database.
UC8	Manage Experiments	1. Administrator logs in to the Turbo Consent web application and is directed to the homepage. Administrator navigates to the admin page. 2. Administrator clicks “Manage Experiments”. 3. Administrator adds experiment to visitor, deletes existing experiments or creates new experiment by filling in experiment details and clicking “Create”. Their changes are saved to the database.
UC9	Logout	1. User clicks “Logout”. The user is redirected to the login page and is no longer authorised to access the Turbo Consent web pages.

Table 3: Key Use-Cases

2.6 Core Use-Case Flowchart

The following figure depicts a flowchart of the core use-case of the Turbo Consent web application, UC2, selecting a consent option.

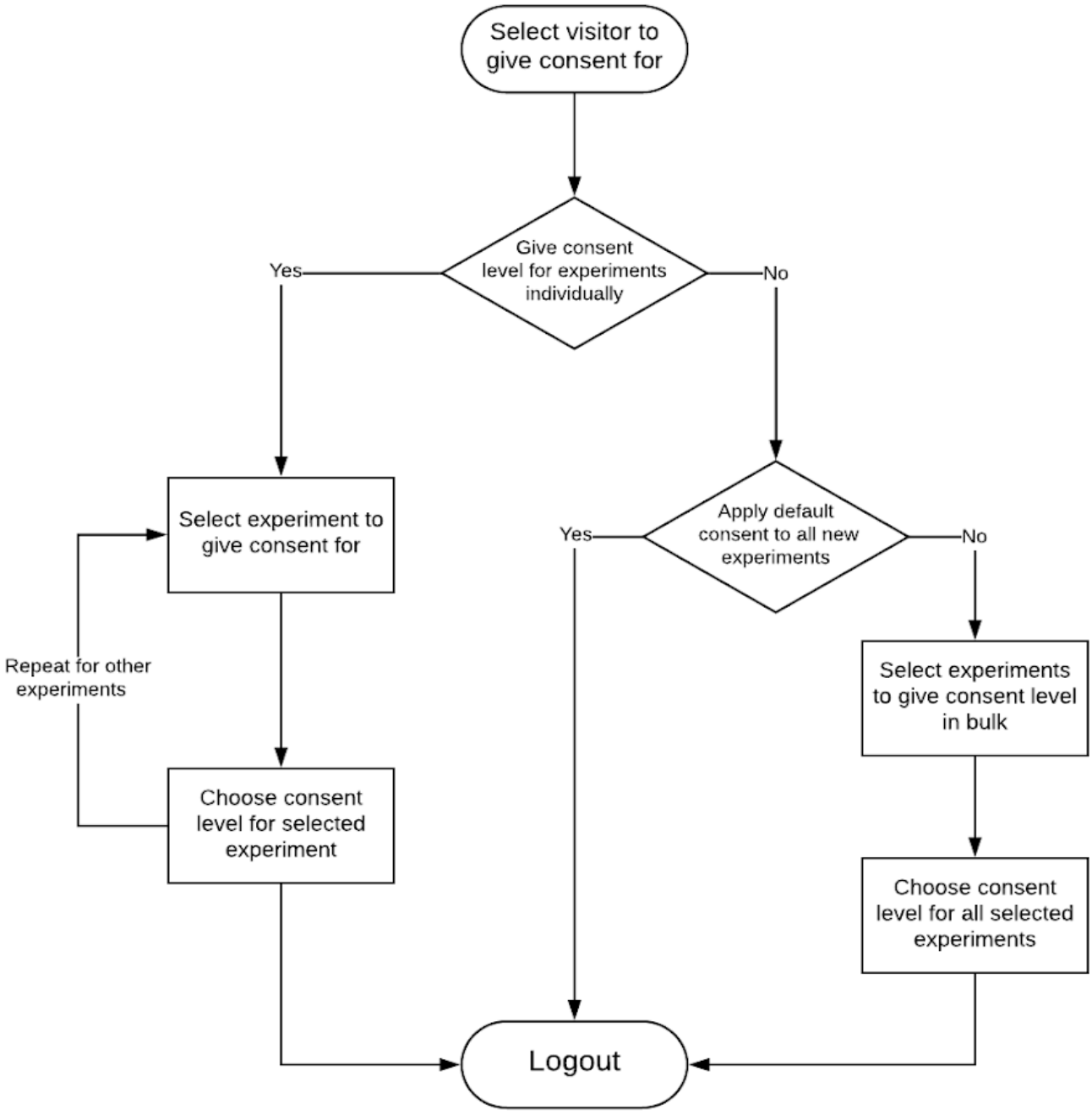

Figure 2: Core Use-Case Flowchart

3 System Architecture

The following figure is a high-level architecture diagram of the Turbo Consent web application.

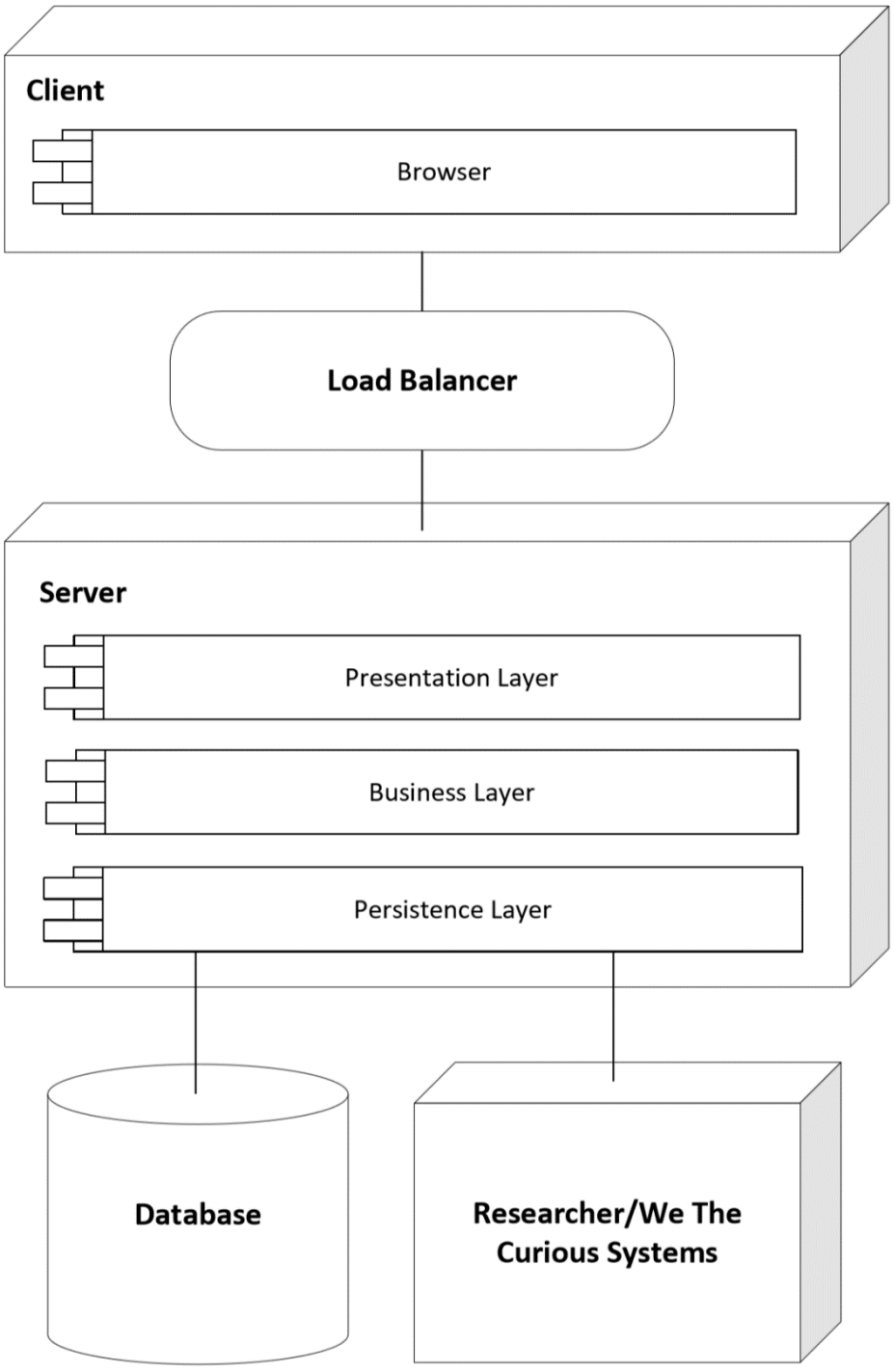

Figure 3: High-Level System Architecture Diagram

The Turbo Consent web application has a system architecture based on the Client-Server Reference Architecture. This is necessary because a client-server architecture is a requirement for browser access. The web application will be deployed onto a Tomcat web server. An Nginx web server will be deployed in front of the Tomcat server to handle HTTP requests from the browser and provide load balancing. This will increase the performance and stability of the Turbo Consent web application even with high-user traffic.

As we are using a Java Web Application Framework, our web server will use a Layered Architecture Pattern, consisting of three layers:

- The **presentation layer** - this will include our UI and presentation logic components.
- The **business layer** – this will include our business logic, business workflow, business entities and potentially an application façade. The logic behind accessibility, security and authentication happens in this layer.
- The **persistence layer** – this includes the DAO (Data Access Object) presentation, allowing persistent data from the database to be presented at the application level. This layer also includes the service agent component which will allow researchers' systems to retrieve data from our database and operate with our web application.

We will be using an Oracle Cloud Database to store all of the Turbo Consent application data.

4 Testing Strategy

4.1 Development Testing

Throughout our project we will be doing Test Driven Development to ensure that all of the features of our web application are working as we expect them to. We will integrate JUnit and Mockito frameworks into the project to automate unit tests on the functionalities of the Turbo Consent web application.

Due to the scale of the project and time constraints, we had to limit automatic testing to the critical areas of our web application. Therefore, we decided to write JUnit tests for the business layer of our web application as this is where all of our core business logic occurs that controls the functionality of Turbo Consent. Manual testing is easier to perform on the other areas of the web application for the following reasons:

- The Presentation layer - this layer uses the Spring Framework's Controller classes to map HTTP requests to html web pages (that use CSS and Javascript). Errors are easy to identify because when running the application, it is easy to identify if the webpage has appeared correctly.
- The Persistence layer - all of the data repository interfaces are generated by the Spring Framework automatically and are therefore assumed to work correctly. The only classes in this layer are the Entity classes which only use trivial getter and setter methods so they are also assumed to work correctly.

Table 4 (below) explains the test data used and the expected results of the automatic JUnit tests that are run on our business layer methods:

Business Logic Method	**Method Description**	**Test Data and Expected Return Value**
getAccount()	Return Account object from Account ID.	• Valid **Account ID** must return the valid **Account** with that ID. • Invalid **Account ID** must return **null** object.
getAccountID()	Return Account ID from email.	• Valid **email** must return the corresponding Accounts **ID**. • Invalid **emails** must return **-1**.
getAccounts Visitors()	Return list of Visitor objects linked to an Account from Account ID.	• Valid **Account ID** must return all of the corresponding Accounts **Visitor** objects. • Invalid **Account ID** must return and empty **Iterable** object.
updateAccount Consent()	Update default consent option of multiple Visitors on an Account.	• Valid list of **Visitor IDs** and **Consent Option** must go through and update the Accounts visitors' default consent to that consent option. Then return **true.** • If either list of **Visitor IDs** or **Consent Option** are invalid, return **false**.
getVisitor()	Return Visitor object from Visitor ID.	• Valid **Visitor ID** must return the valid **Visitor** with that ID. • Invalid **Visitor ID** must return **null** object.
updateVisitor Consent()	Update default consent option of a Visitor.	• Valid **Visitor ID** and **Consent Option** must go through and update that Visitors default consent to that consent option. Then return **true.** • If either the **Visitor ID** or **Consent Option** are invalid, return **false**.
getExperiment()	Return Experiment object from Experiment ID.	• Valid **Experiment ID** must return the valid **Experiment** with that ID. • Invalid **Experiment ID** must return **null** object.
getExperiment Consent Options()	Return list of Consent Options of an Experiment from Experiment ID.	• Valid **Experiment ID** must return a list of the **Consent Options** specifically for that experiment • Invalid **Experiment ID** must return **null** object.
getVisitor Experiments()	Return list of VisitorExperiment objects (Experiments that a Visitor has participated in) from Visitor ID.	• Valid **Visitor ID** must return a HashSet of all of the **VisitorExperiment** objects that that Vistor has completed. • Invalid **Visitor ID** must return an empty **HashSet** object**.**
getExperiment Consent()	Return current Consent Option of an Experiment from Visitor ID and Experiment ID.	• Valid **Visitor ID** and **Experiment ID** must return the **Consent Option** name, for which that **Visitor** has chosen for that **Experiment.** • Either an invalid **Visitor ID** or **Experiment ID** must return the string **"NULL"**.
Update Experiment Consent()	Update Consent Option of a VisitorExperiment.	• Valid **Visitor ID**, **Experiment ID** and **Consent Option** must correctly update that **VisitorExperiment** with that **Consent Option and** return **true**. • Either an invalid **Visitor ID**, **Experiment ID** or **Consent Option** must update nothing, and return **false.**

Table 4: Development Testing Test Cases

4.2 Release Testing

After initial development, the Turbo Consent web application is tested as a 'Black Box' – a process whereby the software is tested by studying the inputs and outputs of various tests, without looking at the internal workings of the program. Tests are drawn up with inputs and expected outputs. During testing, if any of the actual outputs are inconsistent with their corresponding expected outputs, the application clearly has a defect. Table 5 (below) details the tests that will be conducted to ensure the application functions correctly:

Test Description	Input	Output
Admin creates a new account.	Input name, email address, password. Click 'Create Account'.	New user account added to database. Admin redirected to the Create Account page with confirmation message.
Admin deletes an account.	On the create account page, click delete button next to the account you wish to delete.	Database updates by removing account and any dependent entities. Admin redirected to Create Account page with confirmation message.
Admin creates a new visitor.	Input name, account ID, date of birth. Click 'Create Visitor'.	New visitor added to database. Admin redirected to the Create Visitor page with confirmation message.
Admin deletes a visitor.	On the create visitor page, click delete button next to the visitor you wish to delete.	Database updates by removing visitor and any dependent entities. Admin redirected to Create Visitor page with confirmation message.
Admin creates a new experiment.	Input name, description and any custom consent options. Click 'Create Experiment'.	New experiment added to database. Admin redirected to the Create Experiment page with confirmation message.
Admin deletes an experiment.	On the create experiment page, click delete button next to the experiment you wish to delete.	Database updates by removing experiment and any dependent entities. Admin redirected to Create Experiment page with confirmation message.
Admin creates a new visitor-experiment.	Input account ID and Experiment ID. Click 'Create Visitor Experiment'.	New visitor-experiment added to database. Admin redirected to the Create Visitor Experiment page with confirmation message.
Admin deletes a visitor-experiment.	On the create Visitor Experiment page, click delete button next to the Visitor Experiment you wish to delete.	Database updates by removing visitor-experiment. Admin redirected to Create Visitor Experiment page with confirmation message.
User logs into account	Input email address and password.	User is taken to homepage.
User logs out of account.	Click "Log out" button, on any page.	User redirected to the login page, session ends.
User chooses a Consent Option.	Select checkbox next to a Consent Option on the experiment page.	Database updates unique Visitor Experiment Consent Option. User redirected to the Experiments page, with confirmation message.
User chooses a selection of experiments	On the Experiments page, select all the Experiments they wish to update, click the checkbox next	Database updates all of the selected Visitor Experiments with the selected Consent

Test Description	Input	Output
and a Consent Option to batch update them.	to the desired Consent Option. Click Update.	Option. User redirected to the Experiments page, with confirmation message.
User selects consent defaults.	On the Settings page, select the checkbox next to the desired Visitor(s) and the checkbox next to the required Consent Option.	Database updates all of the selected Visitor's default Consent Options to the one selected. User then redirected back to the Settings page, with a confirmation message.
User views previously select consent options.	On the homepage, click on the toggle button "Unreviewed Experiments".	The user is shown a list of previously reviewed experiments along with the selected consent option.

Table 5: Release Testing Test Case

4.3 Acceptance Testing

During development, we visited We The Curious to conduct visitor surveys to try to identify what features users of our application would expect and how they would like the service to work. Using the information we collected, we decided that the application would have to be user friendly and make it easy to understand all of the content on the web pages. Users would like the web pages to be clear and simple, and the application should be supported on mobile platforms. So that providing consent on Turbo Consent is as quick as possible, users would like a feature that allows them to give bulk consent for multiple experiments at once.

To test the administrative functionality of Turbo Consent with real potential users, we have shown our current application to the We The Curious client and allowed them to navigate through the website and test the administrative features. The client was happy with the usability of the admin and visitor features and thought the user interface was aesthetic and easy to use.

In order to test the general usability of the Turbo Consent web application with real potential users, we will go to We The Curious to conduct usability surveys with WTC visitors. We will ask the visitors to navigate through the web application to select a consent option for a sample visitor (UC2). We have selected this use case as it will be the main purpose of our web application for most users. This use case requires users to navigate through most of the pages on the application, giving us an accurate usability evaluation for the entire application.

5 Object-Oriented Design & UML

Figure 4 (below) shows a static UML modelling aspect of the Turbo Consent web application. We chose this aspect as it clearly shows the three layers of our Spring application, referenced in our system architecture - the Presentation, Business and Persistence layers. The Controller classes make up the presentation layer, the ConsentService class is the business layer and the Dao interfaces make up the persistence layer. This diagram helps us to understand the relationships between these classes and interfaces. The Controllers map HTTP requests to a method within their classes. These methods will perform some business logic on some data using the methods in ConsentService. ConsentService retrieves this data using the Dao interfaces. Once the logic is performed on the data, the Controllers pass the data to a view, so the web page can be displayed.

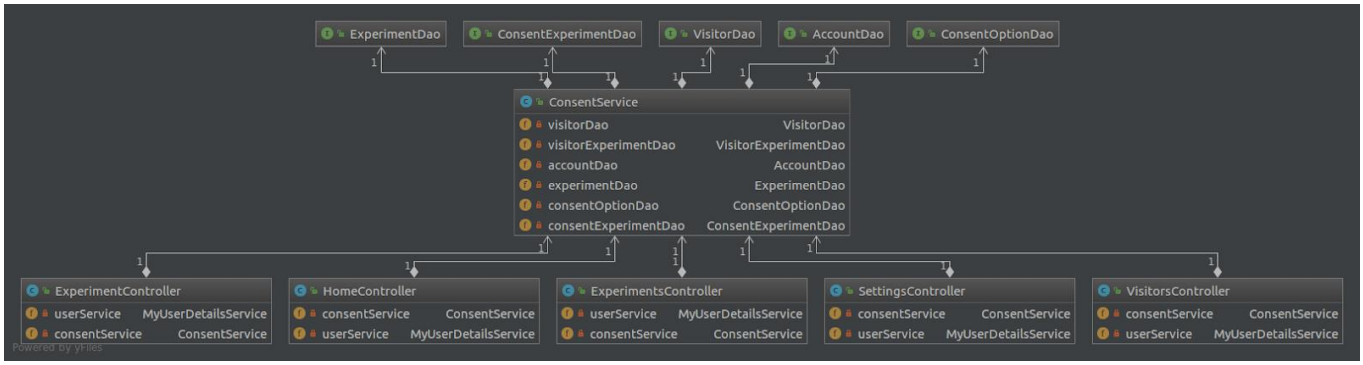

Figure 4: Layered Architecture UML Class Diagram

Figure 5 (below) shows the Entity Relationship Diagram of the Turbo Consent web application. It shows the entities used in our application, the attributes of each entity and the relationship between each of the entities. All of this data is automatically stored and retrieved from a MySQL database during execution of the application.

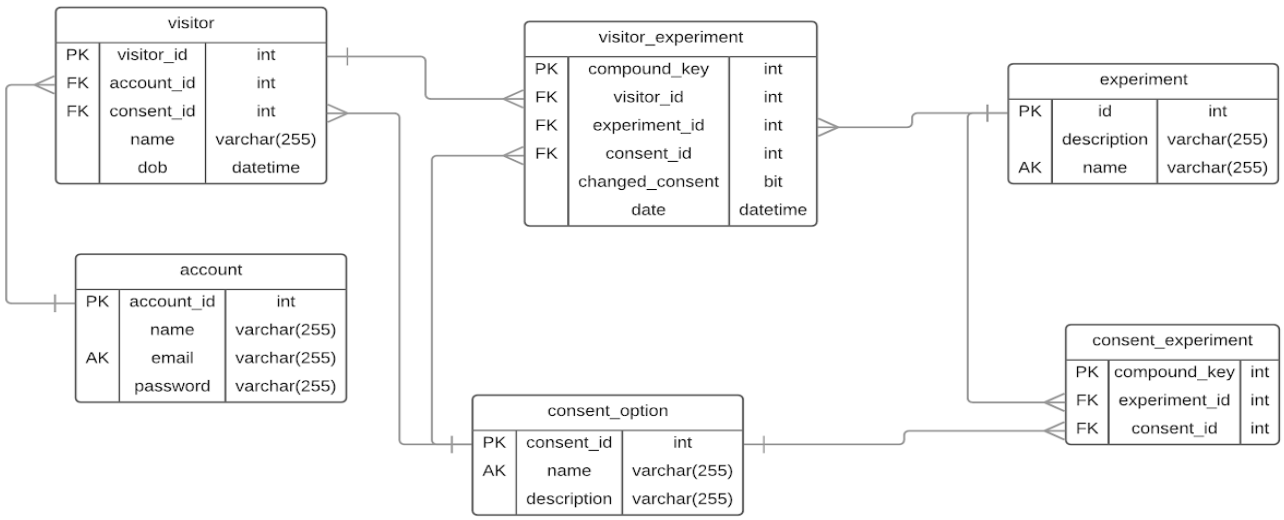

Figure 5: Entity Relationship Diagram

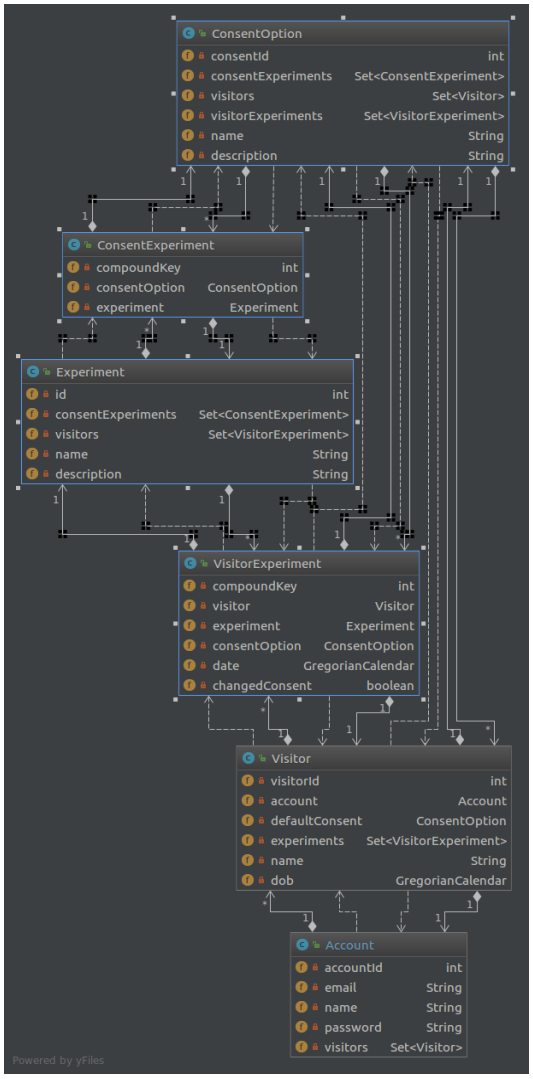

Figure 6: Business Layer UML Class Diagram

This diagram was created with our core use-case in mind (UC2, selecting a consent option). The Turbo Consent web application has two main components – the visitors who participate in experiments and the experiments themselves. Visitors are either above the age of consent, allowing them to make data consent decisions for themselves or below the age of consent, meaning a parent or guardian must make data consent decisions on their behalf. Therefore, a Visitor class will be part of an Account class that has the ability to store multiple visitors and select consent options for each visitor. Parents can have an account that manages the consent options of their whole family. The Experiment class contains experiment information, the available consent options that users can choose from and a list of the visitors who have participated in the experiment. As a Visitor will need to store multiple experiments, there exists a many to many relationship between Visitor and Experiment entities. To normalise this, we use a VisitorExperiment class that stores the associated Visitor and Experiment objects. Similarly, to allow custom consent options for every experiment, we needed to create a ConsentExperiment class to solve the many to one relationship there as well, due to experiments having multiple consent options, and these consent options being reused for multiple experiments.

Modelling our system using UML diagrams has improved our ability to visualise our domain concepts and the structure of our software. The diagrams have given us a clear structure and design of the Turbo Consent web application.

6. User Interface Design

We focused on coming up with a simple, intuitive and responsive design for Turbo Consent. This has been achieved by using the Bulma CSS framework, which allows customisation via SASS. Our main design principle was enabling users to achieve their main purpose of visiting the site, choosing consent levels for new experiments, as quickly and as smoothly as possible.

Figure 7 showcases the wireframe for the web application's initial design, outlining the content that is needed on the experiments page and the consent selection page. Figures 8 and 9 show the experiments page and the consent selection page on the current, operational version of the Turbo Consent web application. Figure 10 shows the mobile view of the homepage and the visitors page.

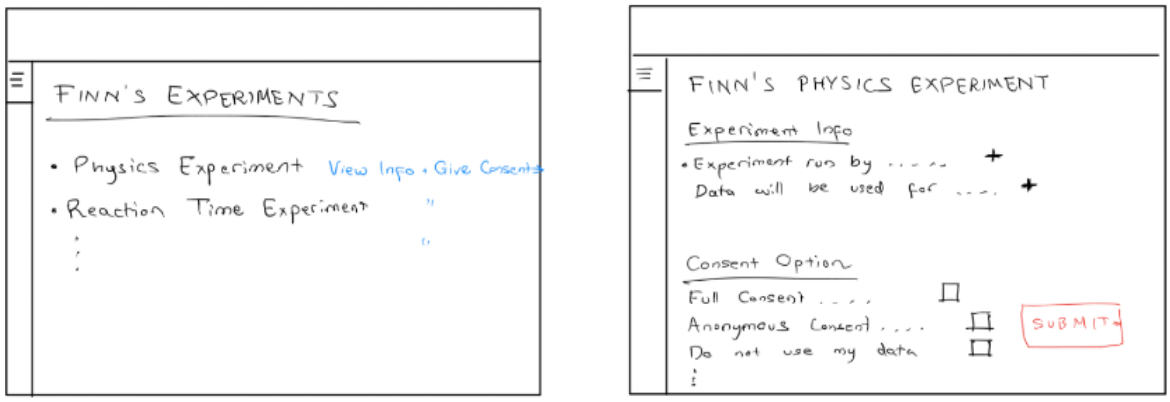

Figure 7: Initial Webpage Design Wireframes

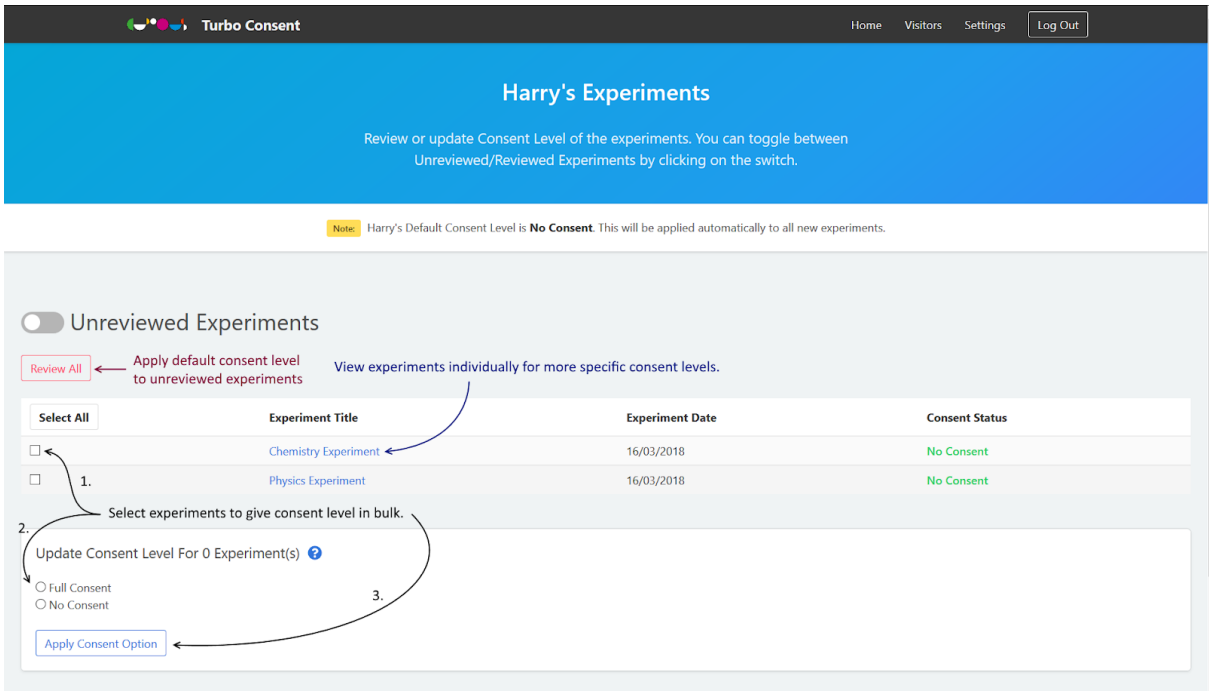

Figure 8: Current Experiments Page

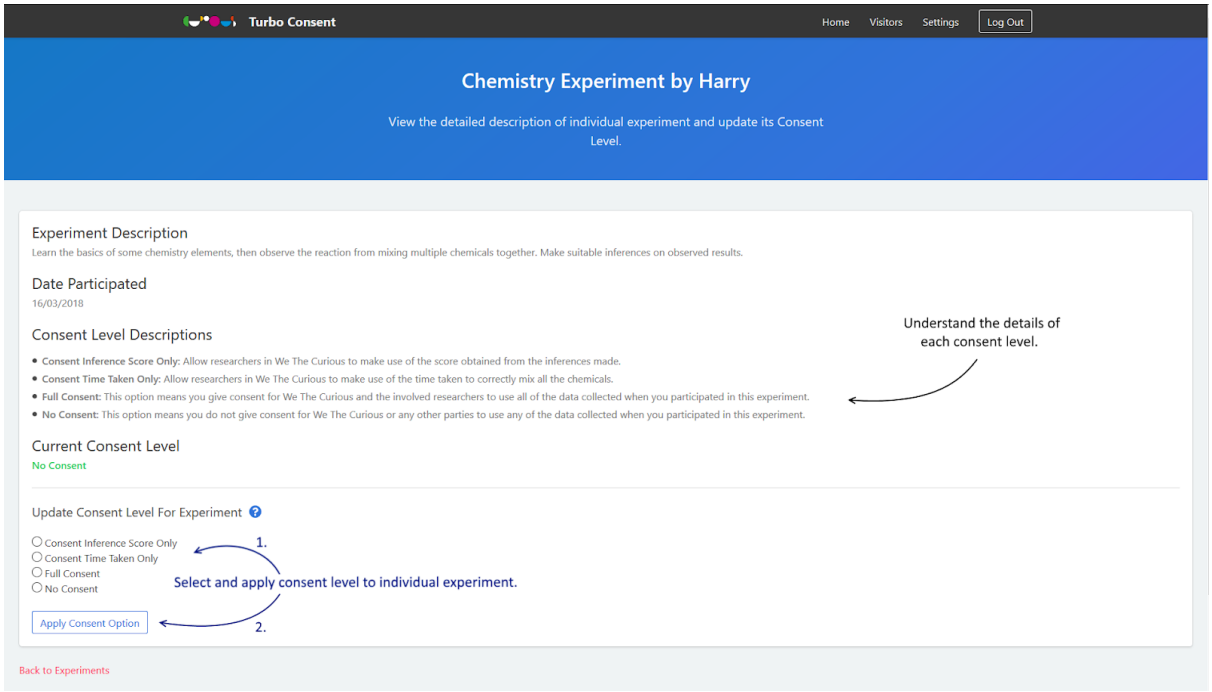

Figure 9: Current Consent Selection Page

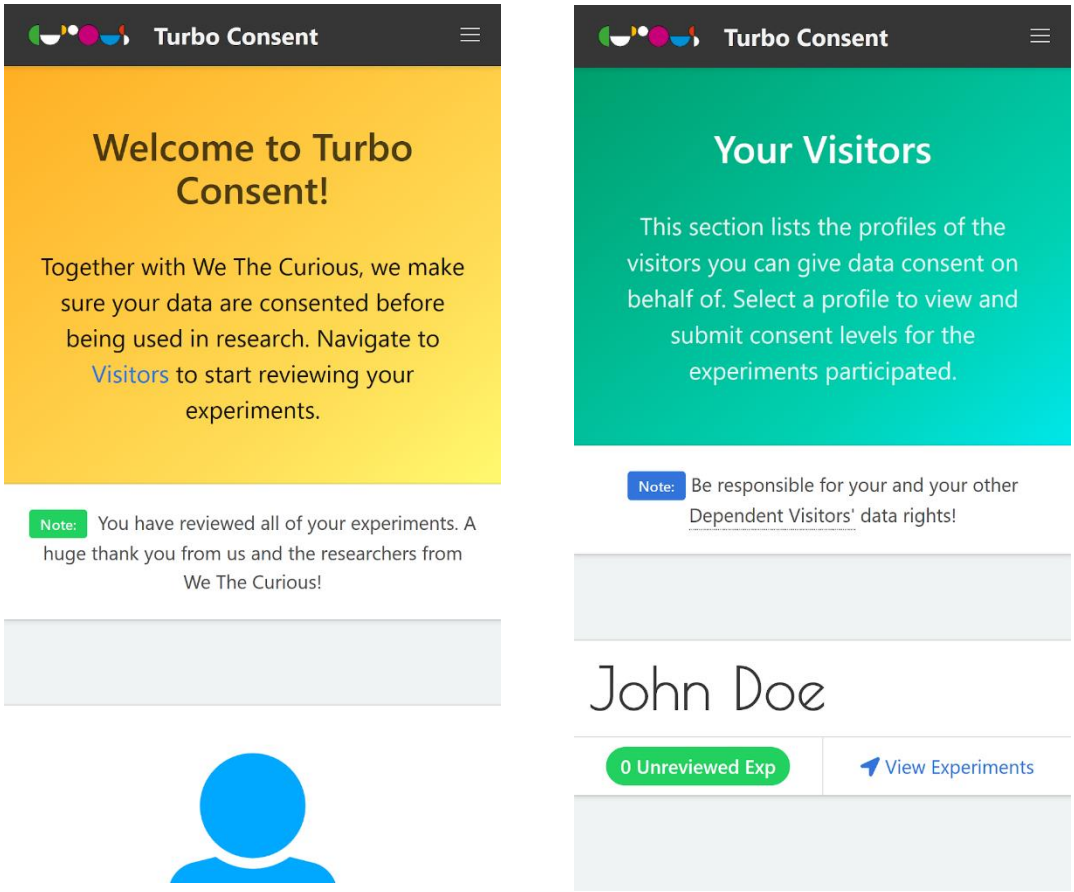

Figure 10: Mobile View of Homepage and Visitors Page