Developing a frontend application using ReactJS and Redux

Developing a frontend application using ReactJS and ReduxDeveloping a frontend application using ReactJS an

Name -Satyam Singh
Roll-No-2101920100246
Branch-CSE
GL Bajaj Institute Of Technology And Management Greater Noida

Developing a frontend application using ReactJS and ReduxDeveloping a frontend application using ReactJS and Redux
Year 2023

Following the rapid growth of social networking applications, Content Management Systems were developed to manage vast amounts of data. This thesis details the development process of building an application that serves as one of these systems. The application is based on ReactJS with Redux as the framework. All concepts and technologies used in the project are explained in their corresponding sections. This includes the relevant theoretical background such as client-server model, Content Management System, and Document Object Model.

The thesis project was commissioned by a mobile application development company called Buddify Oy. The result of this project was a web-based application that the company can use as a Content Management System. The system acts as a centralized control panel and enables the user to manage company product's data.

The main method of development applied in this project was Agile methodology as it is suitable for the changing requirements of the project and for the nature of a start-up company. The Agile is an incremental model, consisting of many iterations. However, this thesis structure follows the traditional Software Development Life Cycle stages to provide an overview of the entire development process.

Keywords: Web application development, ReactJS, Redux

Table of Contents

1 Introduction 4
1.1 Company background 5

1.2 Buddify Content Management System 5
1.3 CMS Users 6
2 Knowledge base 6
2.1 Content Management System 6
2.2 Client-server model 7
2.3 Document Object Model 8
3 Methodology 8
3.1 Software Development Life Cycle 8
3.2 Agile Development Model 10
4 Project objectives 11
5 Requirements 12
6 CMS architecture design 13
6.1 System design 13
6.2 Modular design 14
6.3 Authentication 15
7 Development 18
7.1 ReactJS 18
7.1.1 JSX 18
7.1.2 Stateful components 19
7.1.3 Virtual Document Object Model 19
7.2 Redux 20
7.3 Other technologies 22
7.3.1 HTML and CSS 22
7.3.2 Sass with Compass 22
7.3.3 JavaScript 23
7.3.4 Webpack 24
7.4 Development tools 25
7.4.1 Atom 25
7.4.2 Git and Gitflow 25
7.4.3 Jira Software 27
7.5 Development process 27
7.5.1 Communication 27
7.5.2 Agile Iterations 28
8 Testing 29
9 Project Assessment 30
9.1 Functionalities 30
9.1.1 Authentication 30
9.1.2 User section 31
9.1.3 Posts and comments section 32
9.2 Qualities 32
10 Conclusion 33

Terms and abbreviations

API	Application Programming Interface
App	Application
CMS	Content Management System
CSS	Cascading Style Sheets
DOM	Document Object Model
ES6	ECMA-262 6th edition
HTML	Hyper Text Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
UI	User interface
UX	User experience
XML	Extensible Markup Language

1 Introduction

Following the growth of social networking applications during the last few decades, the demand for solutions to manage vast amounts of data is increasing rapidly. Data management is becoming a key element for many operations in the media industry. Companies have increasingly implemented Content Management Systems (CMS) as a tool to manage various processes of content creation and distribution. These systems can range from simple file management applications to sophisticated systems handling all types of data and integration with a wide range of platforms.

This thesis covers the process of developing one such system. Commissioned by Buddify Oy, Buddify Content Management System was developed for internal use. The system acts as a centralized control panel and enables the user to manage company product’s data. This will help their app meet requirements from app stores, maintain users thus increase marketing and revenue.

1.1 Company background

Buddify Oy is a technology start-up company located in Helsinki, Finland. It primarily operates in mobile application development area, particularly in social networking applications. Buddify currently has 3 apps in development with plans to continuously expand.

1.2 Buddify Content Management System

The method used to manage data prior to the CMS was to access the database tables for modifications. As each user tables contain many attributes with long URL links make it difficult to find the correct person. This has proven to be highly inefficient and have the great potential for human error. Furthermore, Buddify intends to hire third party employees for the management tasks. With full admin access to the database, there is a high risk of security as that user can have access to confidential information. The users also have the ability to shut down the entire database or other disruptions. Therefore, Buddify required a better solution to manage data and thus, the Content Management System was made.

Buddify Content Management System is the name given by the company for this project. The system is a web application, used as platform for the company employees to manage data that is related to various company products.

The following features are required for the CMS:

- Centralized ad network tracking panel: A panel that keeps track of all advertisement network payments within a selected time frame.

- User management system: A system used to search for users within the selected application. Those users can then be deleted or banned.

- Post and comment management system: A system where posts and comments can be found on different feeds or within the user's profile. They can then be selected and deleted.

- Centralized notifications: A record of all notifications. This includes system errors within backend/frontend applications, database traffic, and usage.

- Messaging application: An instant messaging application

CMS features can be added or changed according to the company needs. However, this project will only cover the user, post and comment management systems.

1.3 CMS Users

There are three different tiers of users, each with different access privilege. In descending order, they are administrators, moderators, and members.

The user with the highest privilege is the system administrator given to the founders of the company. Administrators have access to data from all three mobile applications. In addition to moderator's powers, administrators have authority to assign moderators to the corresponding app and to promote or demote CMS users to any role.

The moderator role is given to employees to manage the data within the assigned environment. They have access to delete functions and can remove application users or any of their posts and comments. Moderators are also permitted to ban users for a certain time duration.

Members are only allowed to view the data within the application they are signed up in. Members cannot modify or delete any data shown on the CMS.

2 Knowledge base

This section provides clarification for all unexplained concepts and theories used in the project.

2.1 Content Management System

The term content refers to any type of audio-visual, visual, sound, or textual information. This can be represented by pictures, sounds, text, video, etc. Content can also be metadata. Metadata can be classified as data giving a description of the subject at matter, data describing formats, parameters and other specific information, or data describing location and condition of the carrier. Data such as title, subtitle, keywords, category, owner rights are some examples of metadata content. (Mauthe 2005)

A system that manages the mentioned content is called a Content Management System. CMS provides a frontend UI that allow users to read, add, modify or delete content without directly interacting with the database.

2.2 Client-server model

Client-server model is an application structure that divides tasks and workload between a service provider and the service requester. The former is called Server, or backend, while the latter is called the Client, or frontend. Clients and servers communicate through a computer network by exchanging request, response packages.

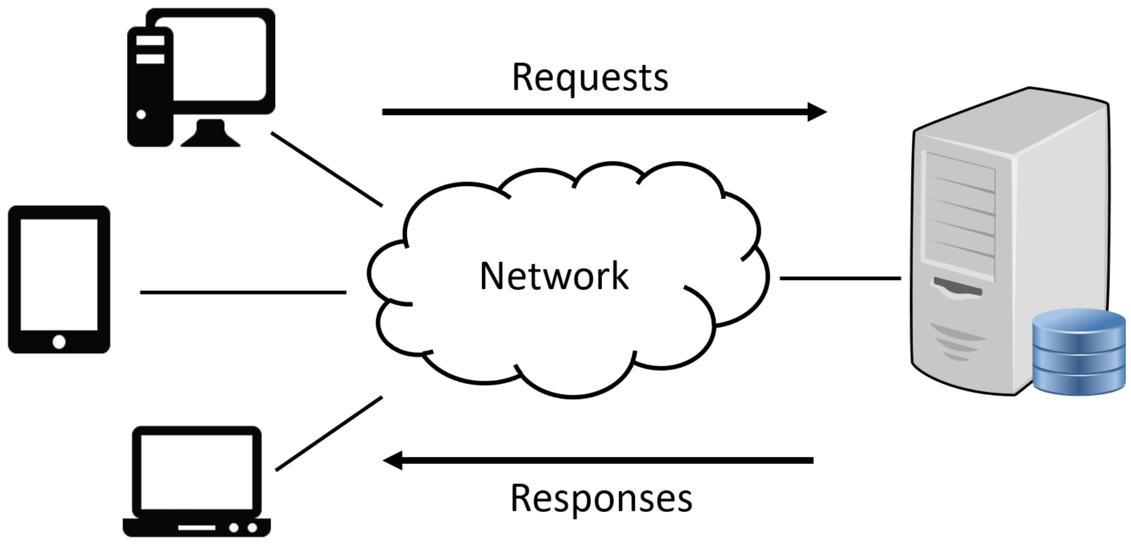


Figure 1: Client-server model

With this model, the client side does not have to be concerned with sharing any data or interact with the database. All resources can be focused on optimizing the user interface and user experience. The server side will handle calculations, business logic and database interactions.
They are dedicated to handle many requests from one or many different clients.

However, as demands for applications with adaptive interfaces and complex user interactions rises, application logic is moving more towards the client. This is because it typically results in a better experience for the user by avoiding full page reloads with every interaction. (Lindley 2014)

2.3 Document Object Model

The Document Object Model is a programming interface for HTML and XML document. It connects web pages to scripts or programming languages. The DOM can be considered as a representation of the page for programs to modify the document structure, style and content. The document is presented in a logical tree with each branch of the tree ends in a node, and each node contains objects. Nodes can have event handlers attached to them. Once an event is triggered the event handlers get executed. (MDN web docs)

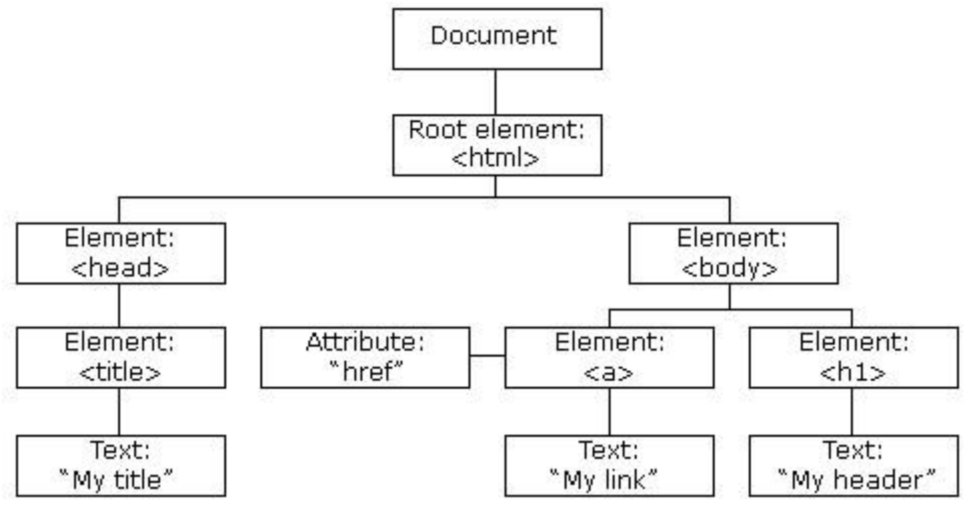


Figure 2: The HTML DOM tree of objects (W3schools)

3 Methodology

Agile Development Model was used as the framework for the development of the CMS. However, the structure of this thesis is based on Software Development Life Cycle (SDLC) to provide an overview of the project.

3.1 Software Development Life Cycle

Software Development Life Cycle, or SDLC, widely used in the software industry as a framework to define tasks performed in each step of the development process. The goal of SDLC is to provide the best product with the resources given.

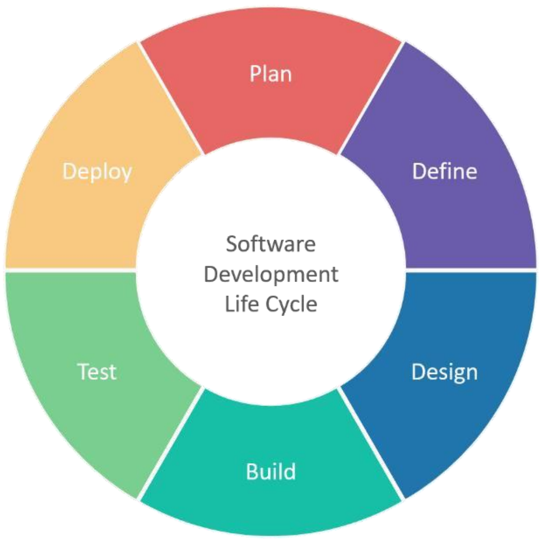


Figure 3: Software Development Life Cycle

A typical Software Development Life Cycle consists of the following stages:

- Planning and requirement analysis: Based on the objective of the product, a project plan will be conducted. The product feasibility will be assessed according to economic resources and the technical requirements. The quality assurance requirements for the product will then be set. Risk assessment is also done in this stage to ensure the project proceeds with minimum risks.

- Defining requirements: Product requirements are to be defined and documented clearly in this stage. This is done through a Software Requirement Specification (SRS) document.

- Designing the product architecture: Base on the SRS document, a Design Document Specification (DDS) document is formed. It contains design proposals for the product architecture including the flow of data with external, and third-party modules if such modules are used. The document will then be reviewed by important stakeholders to decide on the best approach.

- Developing the product: This is the stage where the product is built. The programming code is generated according to the DDS document. The programming language is chosen depending on which type of software is being developed. Coding guidelines are defined by the developer’s organization.

- Testing the product: In this stage, the software is tested for defects. Product defects are reported, tracked and fixed. The testing phase is repeated until the product reaches the standards defined in the SRS.
- Deployment and Maintenance: After the product is carefully tested, it is ready to be deployed. The product is deployed according to the strategy of the organization. Maintenance is done after the product is released when new errors are discovered or when enhancements are needed.

(Tutorials point)

3.2 Agile Development Model

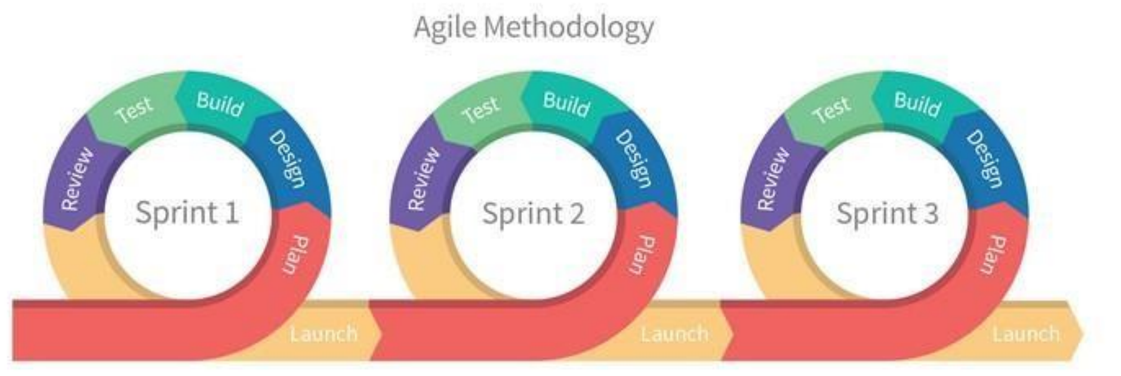


Figure 4: Agile Methodology (Ivanecky 2016)

Agile Development Model is one of the two most popular SDLC models, with the other being the Waterfall Development Model. In opposition to the Waterfall method, the agile method focusses on process adaptability and customer satisfaction by rapid delivery of working software product.

The Agile model is an incremental process model with multiple iterations. Each iteration consists of the fundamental SDLC phases with minor modifications. These phases are planning, designing, building, testing, reviewing and launching the product. A typical iteration can last from one to three weeks. Every iteration involves cross-functional teams working simultaneously in their own areas. (Tutorials Point)

Agile uses an adaptive approach where there is no detailed planning. It only clarifies on what feature need to be developed next and what task is required to complete that feature. Development teams can adapt to a product with dynamic requirements. The product is tested, reviewed by peers and customers, thus greatly reduce the risks of any major failures. (Tutorials Point)

The rationale for selecting Agile as the development methodology is due to its flexible nature. The Buddify CMS project has a dynamic requirement as new functions can be added at any time. The company can also shift feature prioritization causing the development team to change tasks.

Another reason for using the Agile model is that Buddify is a start-up company which means that everything must be done in haste. The company cannot afford to follow a detailed plan with clear definition of each stages as many of the other methodologies require. This can contain high amounts of risk and uncertainty. Instead, they need to push a product out fast to gather feedback then develop further accordingly. Buddify CMS is not published for public use but it is heavily dependent on the backend. If the backend developers are occupied with a different task, the CMS development progressing is halted.

Additionally, Buddify development teams work in close proximity and regularly exchange ideas and discussions. This allows progress to be done without much documentation. Revision and group meetings are also easily arranged.

4 Project objectives

This section details the planning stage of the Software Development Life Cycle.

The objective of the project is to develop a web-based content management system for Buddify Oy’s employees to manage their mobile application content. The application must achieve the predefined requirements in terms of quality and functionalities. Additional functions can be added at any given time if necessary.

The core functionalities of the system include:

- Authentication: Users are required to have an account to be issued an access token and a refresh token. Access tokens are used as a Bearer credential and transferred through an HTTP Authorization header. Excluding login and sign up, access tokens are mandatory for every backend API endpoint requests. Refresh tokens are used to renew access tokens when it expires. In addition, sign up functions are available within the CMS and all tokens are stored within the client’s browser for convenience.

- App user exploration and flagged users: Users can be found on the CMS in the same way as on the mobile application with additional filter options. Flagged users are also displayed on a separate section. CMS moderators and administrators can then choose to delete the user or ban them. Ban durations are chosen by selecting from a few preset options. In addition, IP ban can also be set but should only be used with careful consideration.

- User posts and comments: User posts and comments are displayed using the same UI and filters as on the mobile application. Posts can be found on the main discovery feed, and on user's profile section. With each post and comment, a delete option is provided. All deletion need to be confirmed and are permanent.
- User information: User profile is also provided in the CMS. All information category is displayed according to each app. Report count is also shown here.

The quality requirements of the system are:

- The system must have an uncluttered, simple user interface with readable font and font-size.
- It must be responsive on both the web browser and the mobile browsers.
- Loading indicators are added for asynchronous items.
- There must not be any shuttering while scrolling and the loading speed of each view are reduced to a minimum.
- The system must not cause any crash or freezes to the browser.

5 Requirements

The following section defines the requirements for the CMS. This is the Defining stage of the SDLC.

Buddify CMS is designed to manage content from multiple apps. Each app can contain similar features or entirely different ones. The CMS is therefore required to be flexible and adapt to functionality changes depending on which application the system is managing. Additionally, new features are continuously built as further demands for the CMS and the amount of Buddify mobile application rises. To account for the future growth, extensibility and reusability will be the focus of this project.

Extensibility is a design principle that takes future growth into consideration with each implementation. In software engineering, not everything can be designed in advance. Embracing this, extensibility design emphasizes properties that minimize the effort required to modify or extend the system. The three most important properties of extensibility are modifiability, scalability, and maintainability.

Modifiability is the ease with which a software system can be modified. Modifiability can be determined by how each functionality is separated and architecturally organized, and by how coding techniques are applied. A system has high modifiability when changes to the system require as few changes in each related component as possible. (Bass et al. 2003).

Maintainability is the effort required to locate and fix an error in an operational software. Maintainability is similar to Modifiability in definition. The essential difference between them is that maintainability is also takes the correction of bugs into consideration whereas modification is not.

Scalability as defined by Bondi is “the ability of a system to expand in a chosen dimension without major modifications to its architecture” (Bondi 2000). Scalability can be measured by how the system can add new functionalities or new components with minimal effort, and without having to undertake a major redesign.

6 CMS architecture design

This section details the Designing step in SDLC. CMS architecture is constructed based on the requirements defined previously. Fundamentally, it follows the role of a client-side application and uses React-redux as a framework of development.

6.1 System design

System design for this project is based on the conventional client-server model. Buddify Content Management System sends requests with the same method and headers as the mobile application.

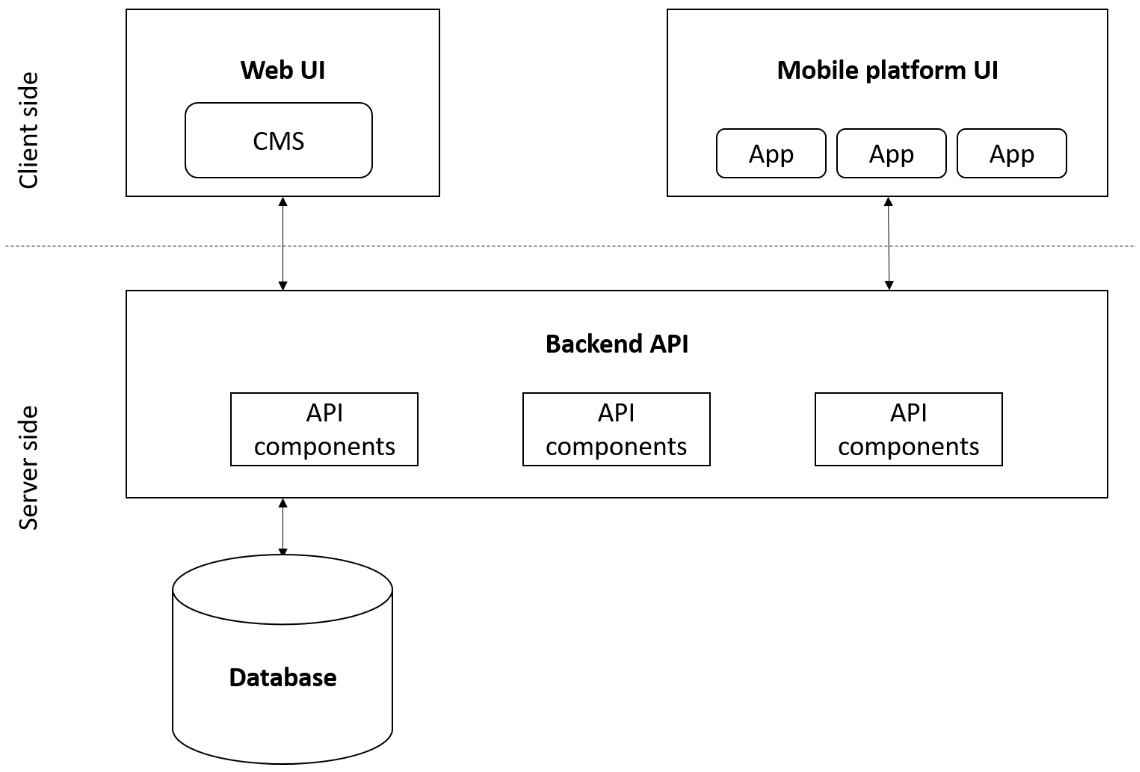


Figure 5: Buddify CMS system model

The backend API handles requests from both the CMS and all mobile applications. It sends back response in identical format with some exceptions where the endpoint is CMS or mobile app exclusive. Calculations, business logic and database interactions are handled by the backend. Frontend applications do contain calculations and structured ways to store data. However, it mainly focusses on providing the best UI with optimal performance and user experience.

6.2 Modular design

Modular design is a design approach that subdivides a system into various independent modules. A modular system provides low coupling, high cohesion components, thus increase flexibility, modifiability, scalability and reusability. From different flow path analysis, it is possible to identify critical components in functionality and performance. By optimizing those components, or replacing them with higher performing ones, it is possible to improve efficiency and performance. (Bejan 2006)

To partition the system into discrete modules, the first step is to describe the different functions the system possesses to separate main components based on functionality. Use case diagrams were used to analyze different flow paths and identify these critical components. Use case analysis is an important and valuable requirement analysis technique that has been widely used in modern software engineering. With its inherent iterative, incremental and evolutionary nature, use case also fits well for agile development.

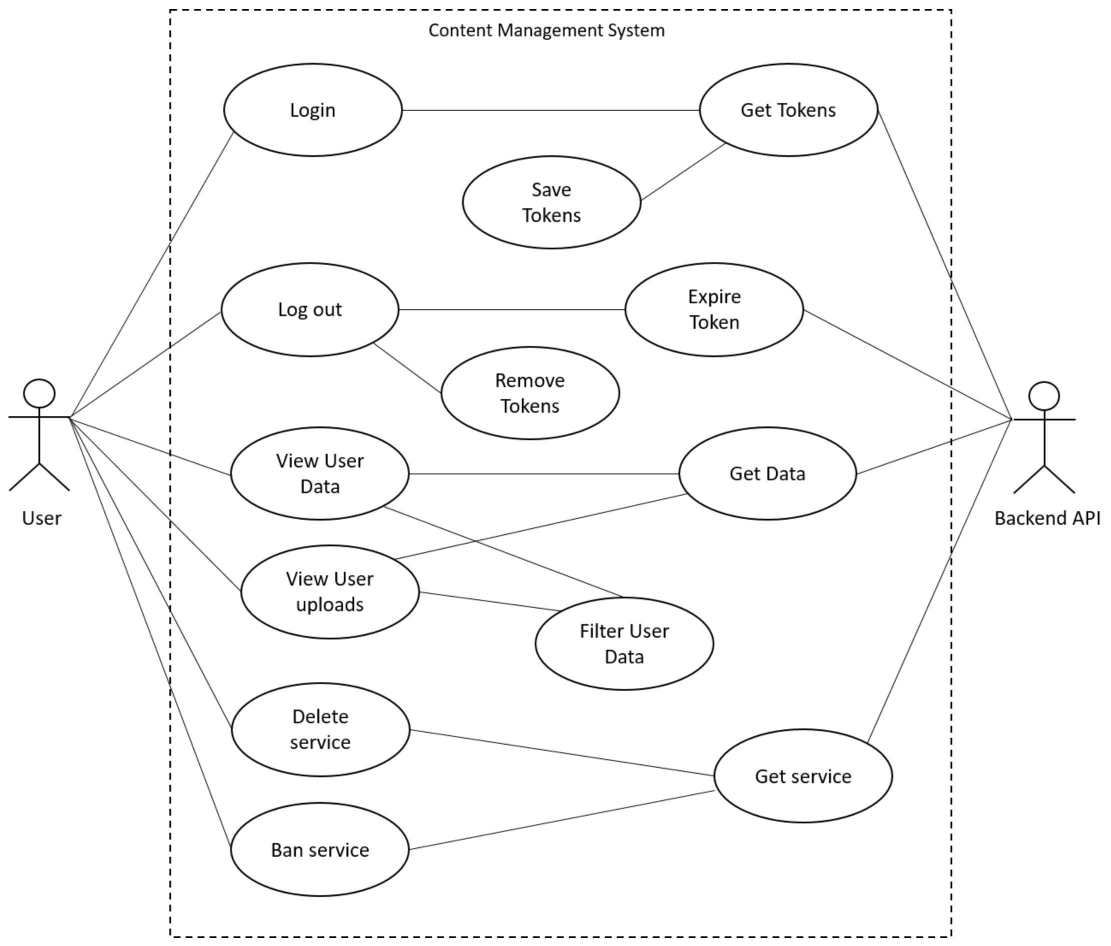


Figure 6: Top level use cases for Buddify Content Management System

The figure above describes the most basic interactions between the CMS user and the backend API. From this diagram, Buddify Content Management System module components are sorted base on the service it provides. They are separated into authentication, user, user uploads, ban and delete services.

6.3 Authentication

Buddify API authentication is based on OAuth 2.0, one of the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flow for web applications. (OAuth 2018) This section will not cover the rationale for selecting this authentication method nor how it is implemented, as it is backend’s responsibility. However, a description of how the CMS handles OAuth tokens, and the authentication design for the frontend system will be provided.

CMS users are issued an access token and a refresh token after a valid username and password have been submitted. Access tokens are like the user's credentials and are sent like an API key. It allows the application to access a user's data and always come with an expiration time. Refresh tokens are used to retrieve a new access token if the current one has expired.

Excluding the login and sign up requests, access tokens are mandatory for every HTTP request sent to the backend API. The token is placed in the authorization header with the syntax: *Authorization: Bearer [access_token]*. The following diagram shows the process of sending a typical API request in Buddify CMS.

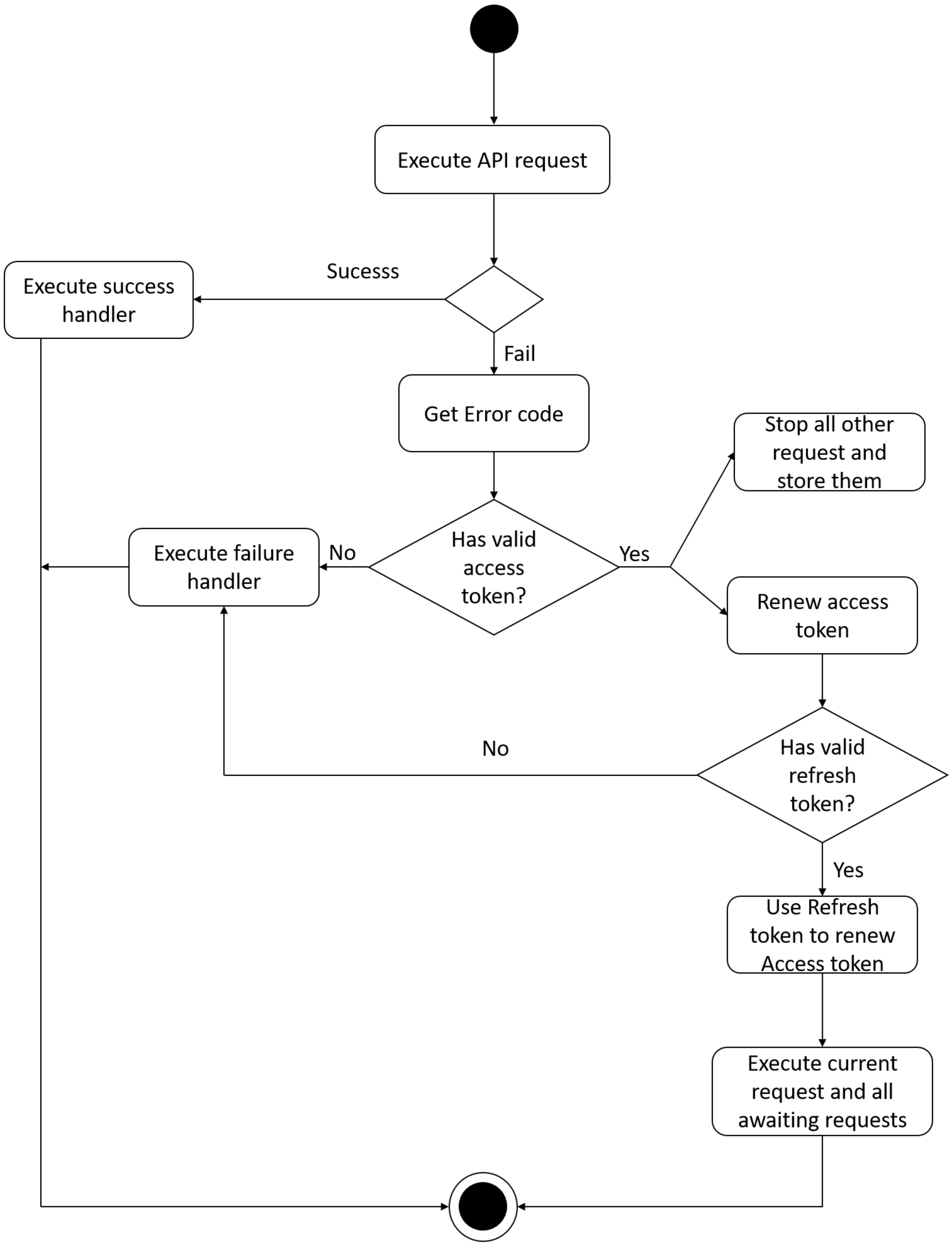


Figure 7: Process of sending a typical API request through the web service module

A request with an invalid token will be caught as an error and the request renew token handler function is executed. In most cases, the entire process completes within milliseconds. However, users can attempt to send additional requests in abnormal cases like slow connection or application suspension. With this in consideration, the handler function first stop all requests that are executed and then store them in order. At the same time, it validates the refresh token. If the refresh token was expired or otherwise invalid, the failure handler function is executed and end the process. If the validation returns valid, the refresh token will be used to generate a new access token. With the new access token, the current request and all awaiting requests queued in this process is then executed.

7 Development

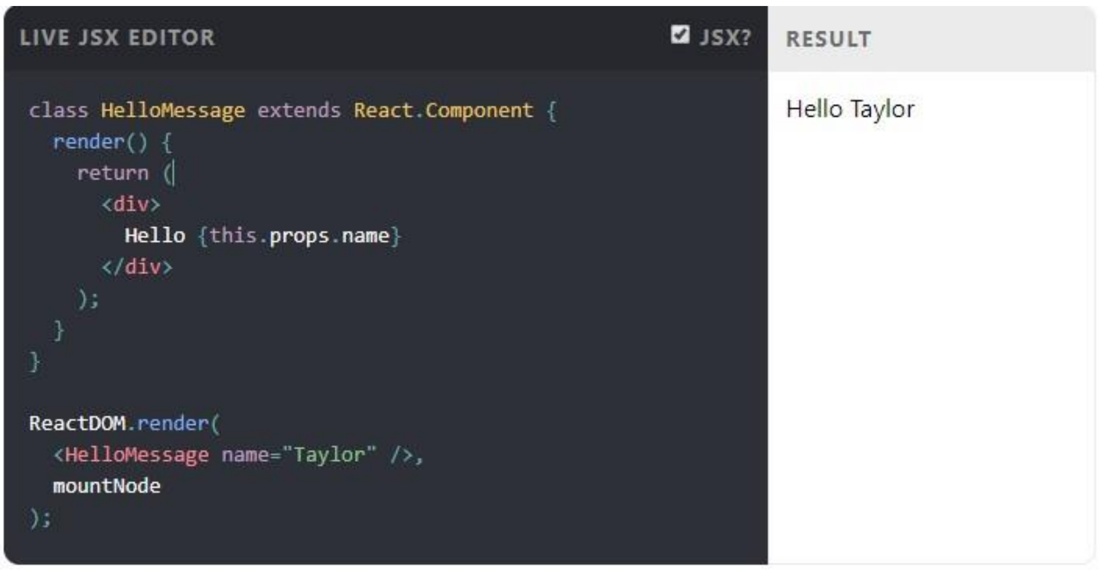
7.1 ReactJS

ReactJS, also known as React or React.js, is an open-source JavaScript library for building user interfaces. It is used for handling view layer in single page applications and mobile applications development. It is maintained by Facebook, Instagram and a community of developers and corporations.

React strives to provide speed, simplicity and scalability. Some of its most notable features are JSX, Stateful components, Virtual Document Object Model.

7.1.1 JSX

JavaScript XML (JSX) is an extension to the ECMAScript syntax without any defined semantics. (JSX 2014) React embraces the fact that rendering logic is inherently coupled with another UI logic. Instead of separating technologies, React uses loosely coupled units called components that contain both. JSX is optional and not required to use React. However, JSX is a good visual aid when working with UI inside JavaScript. It also allows React to show more useful error and warning messages. (React)



```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

Hello Taylor

Figure 8: An example of JSX (React)

7.1.2 Stateful components

React allows users to split the UI into independent, reusable pieces called React Components. React components implement a render method that takes input data and returns what to display. Each component has several lifecycle methods that can be overridden to execute code at particular times during the process. Methods can be called using Reacts API. (React)

State is a plain JavaScript object that is used to record and react to user events. Each class based component defined has its own state object. Whenever a component state is changed, the component, and all of its child components, immediately re-renders. States hold values throughout the component and can be passed down to child components as props.

7.1.3 Virtual Document Object Model

The HTML DOM was originally intended for static pages and thus was not optimized for creating dynamic UI. When the DOM updates, it has to update every node and re-paint the page with the corresponding CSS and layout. It is common for a single page application to contain thousands of dynamically generated nodes that have event listeners attached to them. In dynamic pages, the HTML DOM must check for changes in every node data at a regular interval. This is considerably reduces application performance. The Virtual DOM was invented as a solution to this inefficiency. (Willmott 2017)

The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser. It can be updated without affecting the actual DOM. React has Virtual DOM built in a module called ReactDOM. When updates are supplied, React uses a process called reconciliation, using an algorithm that compares and contrasts changes to know what elements needs updating. React then only change those elements, leaving the others unaffected.

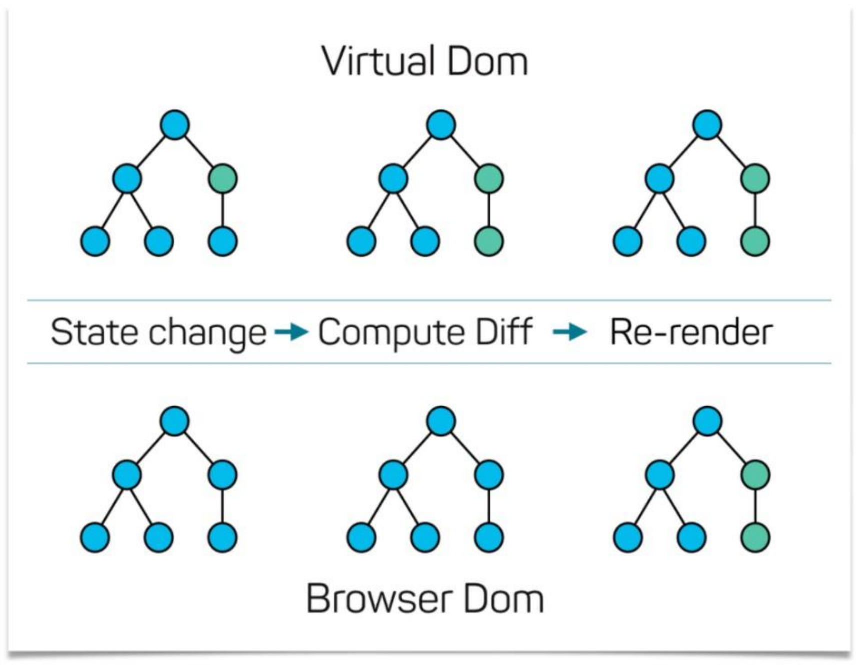


Figure 9: Comparison between Virtual DOM and Browser DOM in different stages of state change

7.2 Redux

Redux is a predictable state container for JavaScript applications. State management is one of the most difficult aspects of software development. State mismanagement is the source of the most errors. Redux is a simplified implementation of Facebook's Flux architecture which is a Model-View-Controller framework. It lessens the complexity by using reducers. Redux reducers are functions without side effects that compute application state. (Geary 2016) Redux is founded on three principles:

- Application state is stored in a single object. Redux stores state in a single JavaScript object to make it easier to map out and pass data throughout the entire application. Centralizing state in a single object also make the process of testing and debugging faster.

- Application state is immutable. In redux, states cannot be modified. The only way to change the state is to provide an action. Actions are immutable JavaScript objects that describes the state changes. Actions are executed in an order to prevent race conditions.

- Reducers specify how the action transform the state. Reducers are JavaScript functions that create a new state with the given current state and action. They centralize data mutations and can act on all or part of the state. Reducers can also be combined and reused.

This architecture greatly increases scalability for large and complex apps. It also enables very powerful developer tools because it is possible to trace every mutation to the action that caused it. With a state and action, the next application state can be predicted with absolute certainty. (Redux)

The following figure shows the data flow of Buddify CMS with Redux.

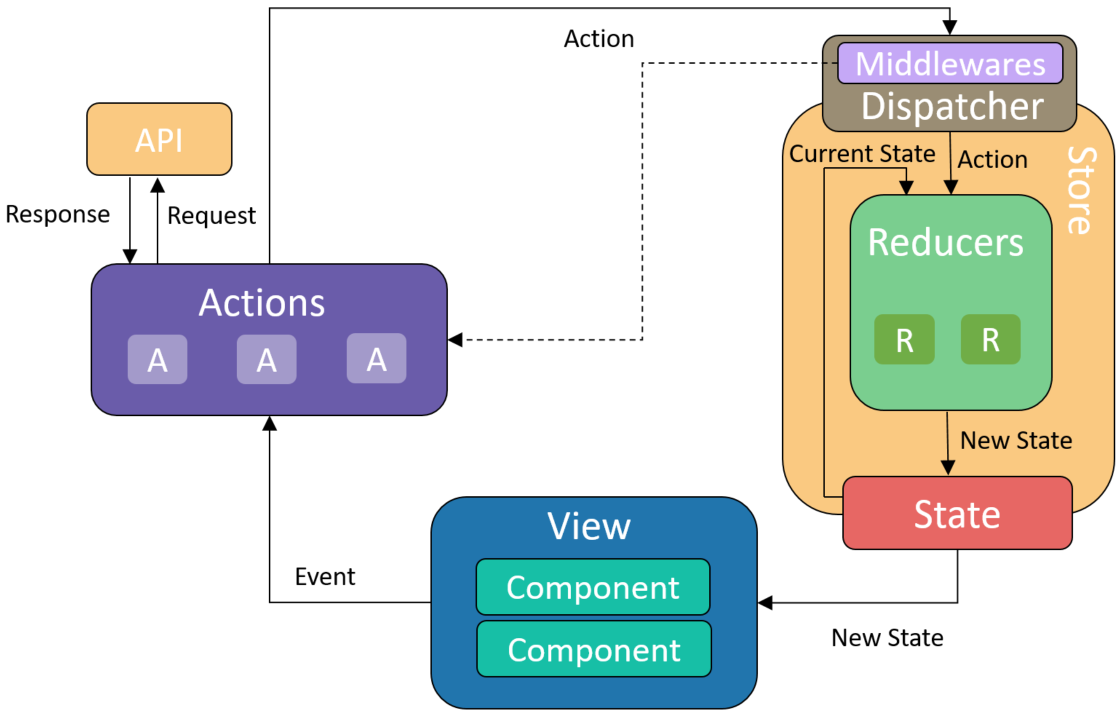


Figure 10: State change process of Buddify CMS with Redux (Terpil 2016)

When a user interacts with a HTML element, the corresponding action calls to dispatch an action. The dispatcher then checks if the action requires a HTTP request to backend API by using middlewares. If such interactions are needed, the dispatcher returns the action and waits for the response from backend as it is asynchronous. Once the request is finished, the action is then passed back. The dispatcher then sends the action and the current state to the reducer. The reducer creates the new state and replace the old one. The redux store with the updated state notifies any view components which the state has change. The component is then re-rendered accordingly.

7.3 Other technologies

7.3.1 HTML and CSS

HTML, or Hyper Text Markup Language, the standard language for describing the structure of web pages. HTML elements are the building blocks of web pages and are represented by tags. These tags label pieces of content. For example, the tag <p> labels the opening of a paragraph with the closing tag </p>. Web browsers do not display HTML tags, they instead only use them to render the content of the page. (W3C 2017)

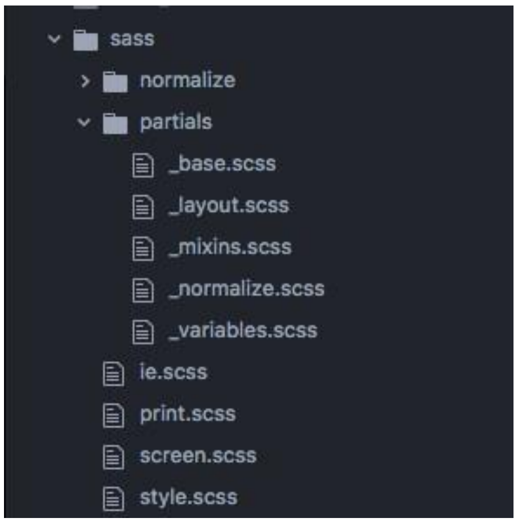
CSS, or Cascading Style Sheets, is a stylesheet language used to describe the presentation of a document written in HTML or XML. CSS describes how elements should be displayed on the web UI. According to the W3C specification, CSS is one of the core languages of the open web and is standardized across browsers. (W3C 2016)

7.3.2 Sass with Compass

Sass is a CCS extension language, providing an alternate way of writing CSS. Sass scripts are written on its' own file under the .scss or .sass extension, depending on which style was used.

Sass allows users to write more maintainable code faster. With plain CSS, elements must be individually target for stylings. The problem with that if the same properties are used for multiple elements, the styles will have to be defined repeatedly. Modification to styles also need to be done in multiple different locations. Sass solves this problem by providing variables, functions and many other tools that allow reusability and increase modifiability.

Compass is an open-source framework built on Sass, designed to make styling with Sass more efficient. It has a large collection of mixins and functions. Compass also provides a compiler. After saving a Sass file, the Compass compiler automatically edit a targeted CSS file, or generate one if no file exists. Configurations such as the name and directory of the file, the root file and the output style can be edited in the compiler config file.



```
sass
  normalize
  partials
    _base.scss
    _layout.scss
    _mixins.scss
    _normalize.scss
    _variables.scss
  ie.scss
  print.scss
  screen.scss
  style.scss
```

Figure 11: Example of Compass folder structure

7.3.3 JavaScript

JavaScript is an interpreted programming language with object-oriented capabilities. Along with HTML and CSS, JavaScript is one of the three core technologies in web development with HTML describing the content, CSS describing how the content is displayed, and JavaScript describing the behaviour of the content. As such, JavaScript is able to run on all modern browsers without any additional plugins or compilers and is used in the majority of modern websites. (Flanagan 2011)

JavaScript provides many good features such as functions, loose typing, dynamic objects, and an expressive object literal notation.

- JavaScript's functions are first class objects with lexical scoping. It has more commonality with Lisp and Scheme than with Java. This makes JavaScript a remarkably powerful language.

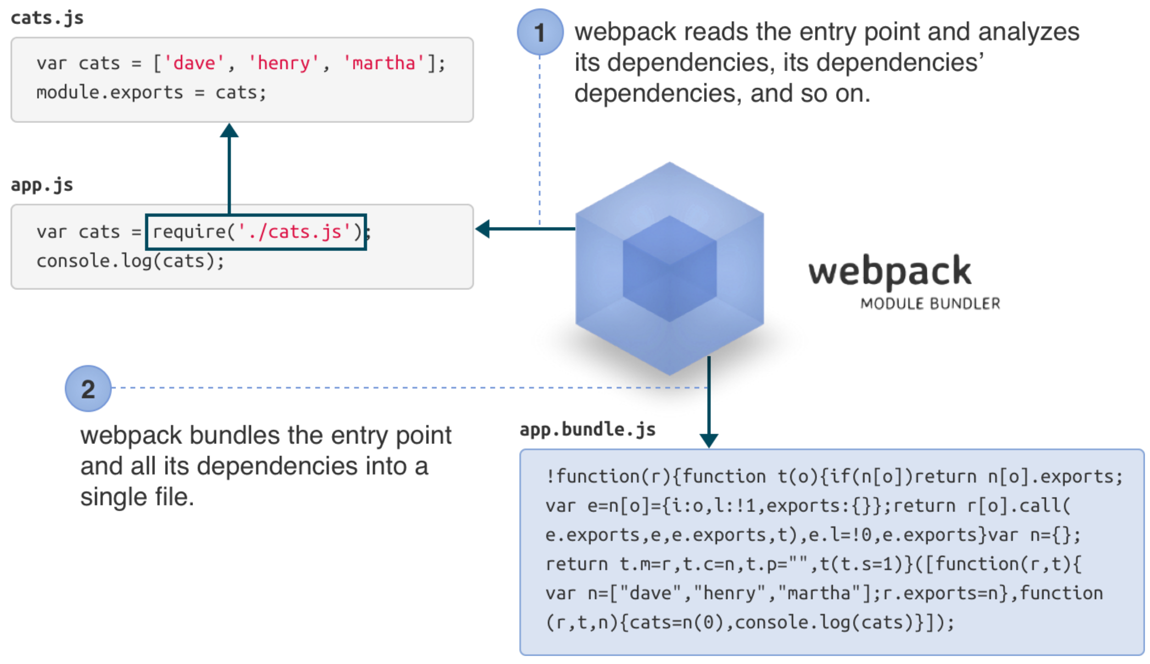
- Loose typing objects liberate developers from having to form complex class hierarchies and reduce concerns about the type system.

- JavaScript has an expressive and powerful object literal notation. Objects can be created simply by listing their components. This notation was the inspiration for JSON, the popular data format.

In conclusion, "JavaScript is a full-featured programming language, as complex as any and more complex than some". (Crockford 2008)

7.3.4 Webpack

Webpack is a module bundler that converts dynamic modules with dependencies into static assets. In application development, it is common to have various modules from either custom files or modules installed by Node Package Manager. These modules can contain one or multiple layers of dependencies. Webpack can bundle everything through a dependency graph, into a JavaScript file. Webpack also allow developers to export and import modules, code, and other assets between JavaScript files. Plugins and Loaders can be installed to extend Webpacks functionality.



cats.js

```
var cats = ['dave', 'henry', 'martha'];
module.exports = cats;
```

app.js

```
var cats = require('./cats.js');
console.log(cats);
```

1 webpack reads the entry point and analyzes its dependencies, its dependencies' dependencies, and so on.

2 webpack bundles the entry point and all its dependencies into a single file.

app.bundle.js

```
!function(r){function t(o){if(n[o])return n[o].exports;
var e=n[o]={i:o,l:!1,exports:{}};return r[o].call(
e.exports,e,e.exports,t),e.l=!0,e.exports}var n={};
return t.m=r,t.c=n,t.p="",t(t.s=1)}([function(r,t){
var n=["dave","henry","martha"];r.exports=n},function
(r,t,n){cats=n(0),console.log(cats)}]);
```

Figure 12: Functionality of Webpack module bundler (GitHub 2017) Webpack

offers the following features:

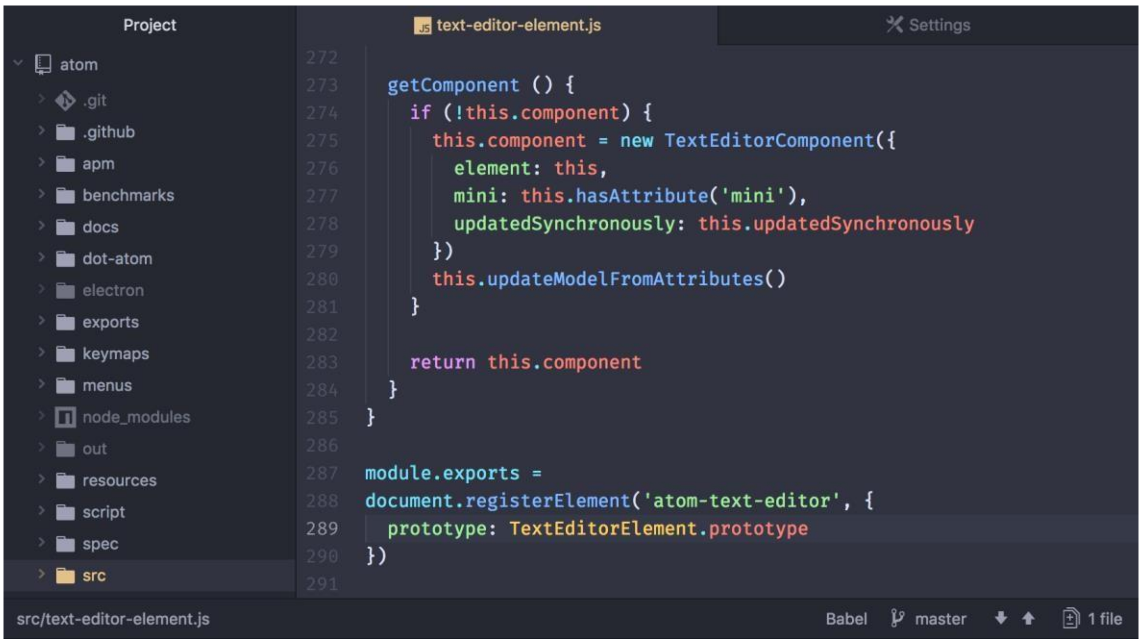
- Dead asset elimination removes assets that are not used to reduce file size.
- Code splitting enables chunks of code to be broken down and used on demand.
- Loaders transform other resources to JavaScript. JSX, Babel loaders are the most important loaders for this project as they are required for React. Babel loaders allow React developers to use ES6 syntax as it will be compiled down to ES5 for full browser compatibility.
- Clever parsing allows Webpack to parse most third-party libraries.

Additionally, Webpack has built in development servers with live reloading. Webpack Dev Server is a small NodeJS Express application that uses Webpack middleware to serve up a bundle for local development.

7.4 Development tools

7.4.1 Atom

Atom is the text and source code editor of choice in this project. The benefits of using atom are that it is a free open source editor with cross platform editing, a built-in package manager and a file system browser with multiple panels. (Atom 2018)



```
  getComponent () {
    if (!this.component) {
      this.component = new TextEditorComponent({
        element: this,
        mini: this.hasAttribute('mini'),
        updatedSynchronously: this.updatedSynchronously
      })
      this.updateModelFromAttributes()
    }

    return this.component
  }
}

module.exports =
document.registerElement('atom-text-editor', {
  prototype: TextEditorElement.prototype
})
```

Figure 13: Atom user interface

In addition, jsLinter can be installed using atom’s package manager. It has high customizable options but as default, jsLinter looks for syntax errors and correct them when the file is saved. It also arranges the code in a standardized coding style such as ESLint, Airbnb, or standard JS. This is greatly increases modifiability and maintainability of the source code.

7.4.2 Git and Gitflow

Git is the most widely used modern version control system by a large margin. It is a powerful tool for a development team to collaborate in a project. With high performance, security and flexibility, Git provides the most optimized service for developers to manage their source code. (Atlassian)

A Git workflow is essential to accomplish work in a consistent and productive manner. It encourages users to leverage Git effectively. Gitflow is a git workflow design that defines a strict branching model based on project release. This provides a robust framework for managing larger projects. (Atlassian)

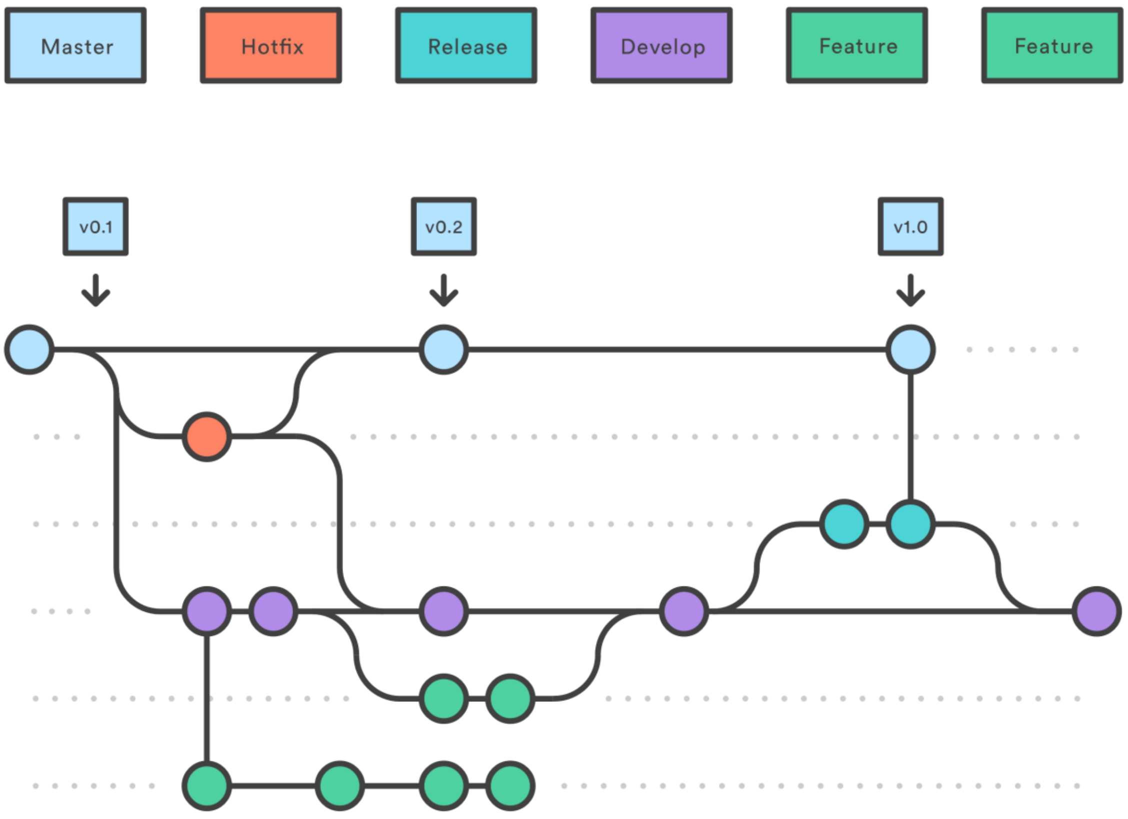


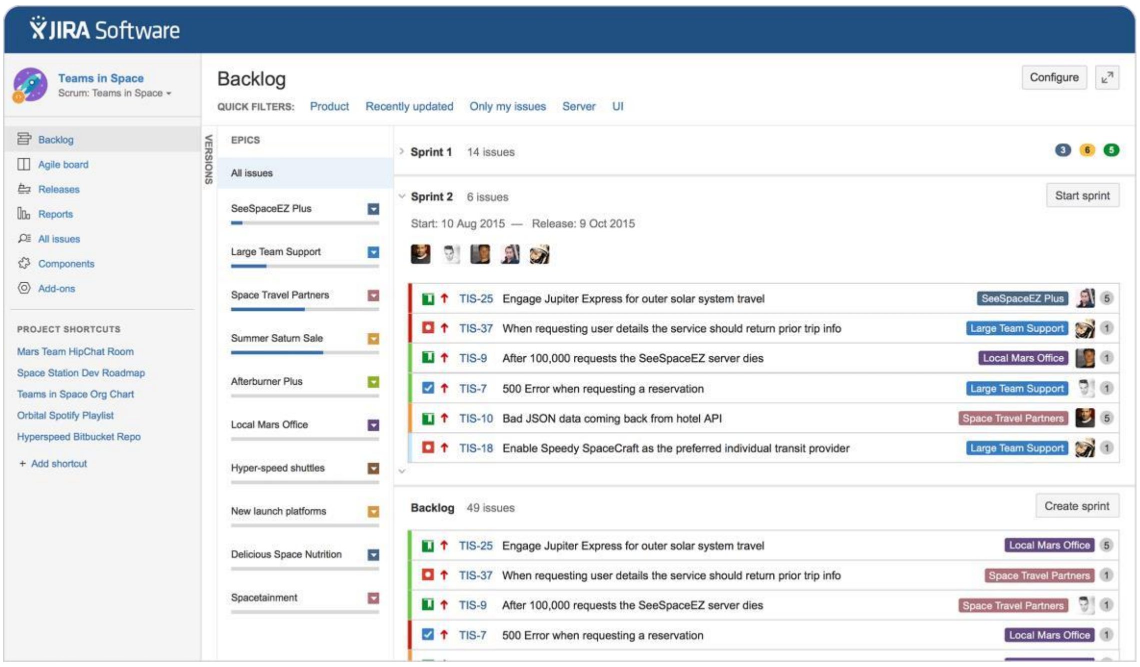
Figure 14: Representation of Gitflow Workflow branches (Atlassian)

All Buddify projects follow the Gitflow Workflow, including the Buddify CMS project. Instead of a single master branch, this workflow uses two branches to record the history of the project. The master branch is the official release history while the develop branch serves as an integration for features. Each new feature should reside in its own branch. When a feature is finished, it is merged back into develop. Features do not interact with the master branch. Once develop has acquired enough features determined by the user or a release date, it will be duplicated to use as the master branch. The develop branch must never be behind the master branch. Hotfix branches are bases directly off master and merged into both master and develop branch after completion. (Atlassian)

Buddify also use the develop branch to testing products within the company team before release. The develop branch is connected to a dedicated server called the Dev server. Buddify developers can test the feature or application on this server without any restrictions as it does not affect the production server. Buddify CMS manages the content from the Dev server's database. Additionally, all feature branches require a code review from one or many assigned reviewers before merging into the develop branch. This code review process is called a pull request.

7.4.3 Jira Software

Jira Software is the most popular software development tool used in Agile development. It provides issue tracking and project management functions. Jira Software's rich planning features, such as Scrum, Kanban, and Mixed methodology, enable flexibility in planning. Jira can be used to keep track of the developing process with accurate estimations that helps the development team become more accurate and efficient. The order of stories, issues and bugs in product backlog can be arranged in prioritization levels. Jira also have extensive reporting functionality that provides critical insight into the agile development process. (Atlassian)



TIS-25	Engage Jupiter Express for outer solar system travel	SeeSpaceEZ Plus	5
TIS-37	When requesting user details the service should return prior trip info	Large Team Support	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Local Mars Office	1
TIS-7	500 Error when requesting a reservation	Large Team Support	1
TIS-10	Bad JSON data coming back from hotel API	Space Travel Partners	5
TIS-18	Enable Speedy SpaceCraft as the preferred individual transit provider	Large Team Support	1

TIS-25	Engage Jupiter Express for outer solar system travel	Local Mars Office	5
TIS-37	When requesting user details the service should return prior trip info	Space Travel Partners	1
TIS-9	After 100,000 requests the SeeSpaceEZ server dies	Space Travel Partners	1
TIS-7	500 Error when requesting a reservation	Local Mars Office	1

Figure 15: Jira Software issue tracker (Atlassian)

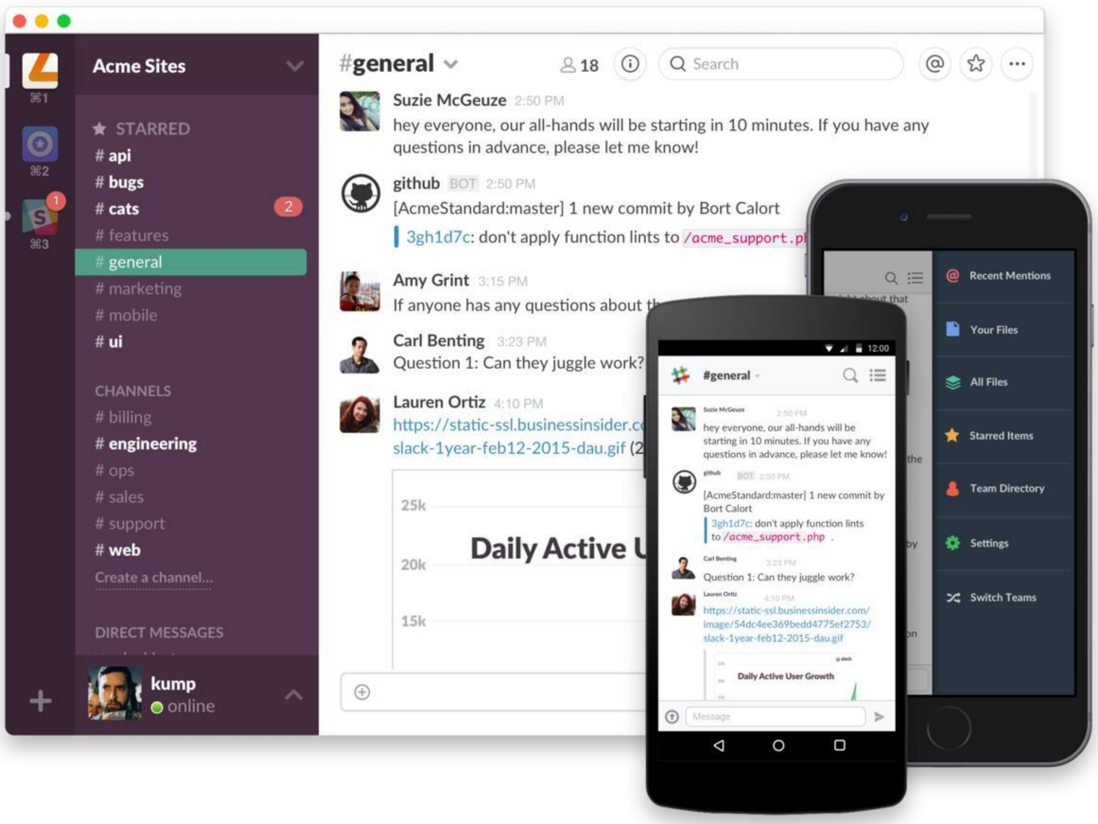
7.5 Development process

This section describes how the Agile development methodology was used in the project. Collaboration tools are also explained briefly in this section.

7.5.1 Communication

Buddify currently has different teams responsible for frontend, backend, Android, and iOS development. Most communications between each member are done through team meetings and through Slack.

Slack is a cloud-based application software designed to help people involved in a common task to achieve their goals. Slack provides a messaging application with collaboration features such as public and private channels. Unlike lengthy email chains, team members can join and leave channels, or set different notification settings as needed. Everything that has been posted in a channel or in direct messages is archived and can be searched even when users have left the channel. Slack also enables file sharing and integrates with a large number of third-party services.



Suzie McGeuze 2:50 PM
hey everyone, our all-hands will be starting in 10 minutes. If you have any questions in advance, please let me know!

github BOT 2:50 PM
[AcmeStandard:master] 1 new commit by Bort Calort
3gh1d7c: don't apply function lints to /acme_support.php

Amy Grint 3:15 PM
If anyone has any questions about

Carl Benting 3:23 PM
Question 1: Can they juggle work?

Lauren Ortiz 4:10 PM
https://static-ssl.businessinsider.com/image/54dc4ee369bedd4775ef2753/slack-1year-feb12-2015-dau.gif

Daily Active User Growth

Figure 16: Slack user interface on desktop and mobile platforms (Slack)

Team meetings are an important part of agile development. These meetings are held once per week and lasts between one to two hours. The purpose of the of the meeting is to plan agile the next agile iteration and review the current one. Each team will also show case their work and the problems they face.

7.5.2 Agile Iterations

Following the Agile Development Model, the project consists of many iterations of planning, designing, building, testing and reviewing. These iterations are also known as sprints. This section will briefly describe the process of a typical iteration.

Each Sprint normally last for one to two weeks. The planning phase is done during the weekly team meetings. The project leader starts with a prioritized product backlog prepared beforehand and discuss each item with the development team. The team will then collectively estimate the effort needed to carry out the task and plan an outline of its completion. Using the Jira Software, a to-do list for the week is then created for each development team.

One or many features can be developed during a sprint. If the feature is complex, feature designs are made and reviewed by other team members. Magic Whiteboards, a whiteboard on a roll that sticks with static to any flat surface, are used to pitch ideas. See Appendix 5: Buddify design plans.

After the design is decided, the development phase commences. To keep everyone informed on the sprint progress across the teams, a daily report is written on a dedicated slack channel. These reports are informal and brief, taking no more than 15 minutes. The contents of the report include what was completed yesterday, a plan of what is done today and a notice if progress is blocked by other teams.

When a feature is finished, all git commits are pushed on to the GitHub repository. Before a merging to the main branch, a pull request is formed and approved by the assigned reviewers. Reviewers are parties that are involved with the feature currently being developed, or some one with knowledge on the same subject. Once a pull request is sent, all reviewers will receive a notification and can then review the set of changes and discuss potential modifications if necessary. If such modification is needed, it must be done before another pull request is created. A feature branch is only permitted to merge when all reviewers approve the request. Agile promotes rapid feedback to make the product and development culture better.
Continuous improvement is what sustains and drives development within an agile team.

Once all reviews are approved, the feature is ready for launch and the next iteration begins.

8 Testing

Testing is the process of verifying that the program is working as intended. In this stage, each feature is checked for defects. Product defects are reported, tracked using Jira and fixed by the person responsible for the tested feature. The testing phase is repeated until the product reaches the standards defined in previous stages.

In this project, Testing is done in two stages: Unit testing during development and after deployed to the Dev server.

During the development process on local server, developers of a feature is required to ensure that the same feature passes all testing scenarios. Testing scenarios are defined during discussions among the development team members. The most common scenarios are slow internet connection, abnormal user inputs and rapid intakes of user input. The tester follows each scenario to check if the feature is producing the expected outcome. Currently, the test done on this stage is performed manually for Buddify CMS. However, unit testing will be automated for new features after the core functionalities have been completed.

After the feature passes tests that are conducted on the client-side, it will be tested on the Dev server. Testers include every member in a development team can test this feature live on the development server. As the server has its own dedicated database, modification to the content here does not affect the mobile application's content. Therefore, testers are permitted to test any feature, including ban and delete features. If any errors or performance issues occur, it will be reported using Jira Issue tracker. This testing phase typically lasts for one week.

9 Project Assessment

This section provides an assessment on the result of the project based on the defined requirements mentioned in the project objectives.

9.1 Functionalities

9.1.1 Authentication

The authentication section has achieved all the requirements mentioned. In addition, authentication was partitioned into a service module in the Actions section. Every request must go through this action.

Additional requirements achieved include:

- Users are required to have an account to be issued an access token and a refresh token. Invalid accounts will show the response message error description. Username inputs remove capitalize letters and spaces when the request is sent.
- Authentication tokens are saved using cookies.
- Signup forms must have the following fields: Username, Password, Email, Full name, Date of birth, Gender, Profile picture, Country, Location. The location section must have a search with auto completion.

Reference Appendix 1: Authentication service

9.1.2 User section

The user section has achieved the predefined requirements with the exception of additional filter options. This includes the application user exploration, flagged user list, and user information. The CMS currently only provide the same options as the ones in the mobile application. Reference Appendix 2: User section

Flagged users are displayed on a separate section using the navigation bar. The template for the flagged users section is identical to the user exploration section.

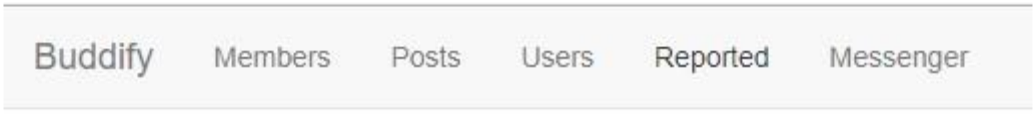
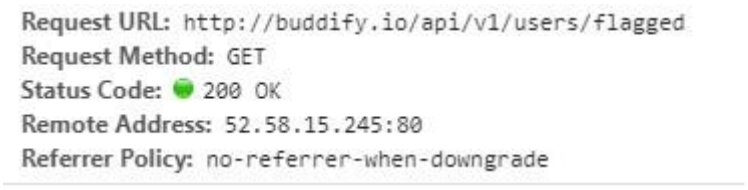


Figure 17: Buddify CMS navigation bar

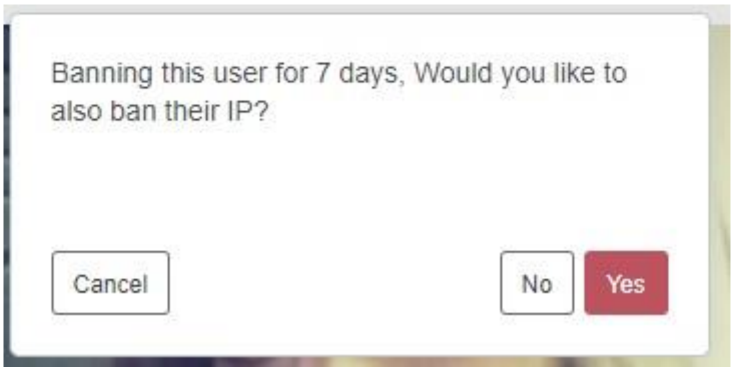
Users that have previously been banned or deleted will still show in this section as backend was required to keep a history of flagged users. As such, UI reference will not be shown for this section as it contains nudity and other inappropriate content, instead the request status will be displayed.



```
Request URL: http://buddify.io/api/v1/users/flagged
Request Method: GET
Status Code: ● 200 OK
Remote Address: 52.58.15.245:80
Referrer Policy: no-referrer-when-downgrade
```

Figure 18: Successful request for flagged users

In addition, the ban and delete panels is provided with a confirmation popup. This is made to reduce the risk of accidental deletion of users.

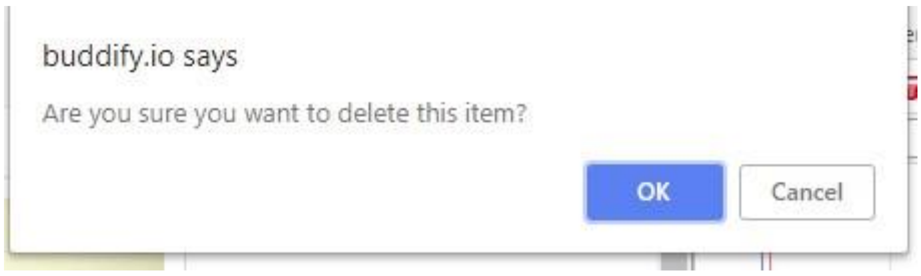


Banning this user for 7 days, Would you like to also ban their IP?

Figure 19: Confirmation popup windows for users

9.1.3 Posts and comments section

User posts and comments are also successfully implemented. User posts and comments are displayed using the same UI and filters as on the mobile application. Posts can be found on the main discovery feed, and on user's profile section. With each post and comment, a delete option is provided. All deletion need to be confirmed and are permanent.


buddify.io says

Are you sure you want to delete this item?

Figure 20: Confirmation popup windows for posts and comments

Reference Appendix 3: Posts and Comments section

9.2 Qualities

The quality requirements of the systems have been achieved. Quality check is done by the project leader with each new implementation.

Buddify CMS has achieved the following quality requirement:

- The system must have an uncluttered, simple user interface with readable font and font-size. UI related requirements are reviewed by testers and changes are proposed through a ticket using the issue tracker.

- It must be responsive on both the web browser and the mobile browsers. A dedicated Sass file was created for mobile view and three different viewports. In smaller view ports, the navigation bar is collapsed, items are displayed in one column, font sizes and button sizes are increased accordingly.

- Loading indicators are added for asynchronous items. Asynchronous items are attached with the *isFetchingData()* method which only resolves when the request is completed. Loading indicators are attached to each of these items.

- There must not be any shuttering while scrolling and the loading speed of each view are reduced to a minimum. The system must not cause any crash or freezes to the browser. React greatly reduces the loading time for each section as it only loads the parts that have been modified. Webpack also bundles modules into a minified file which improves performance. In addition, all unused event handlers are cleared preventing memory leaks that can cause browser freezes or crashes.

Reference Appendix 4: Quality requirements

10 Conclusion

In conclusion, the Buddify Content Management System project was a success. The CMS achieved all the predefined functions and quality requirements. With each implementation, there was considerable effort to enhance extensibility. As a result, the product had high maintainability, modifiability, and scalability. Deployment and maintenance will be done in the upcoming future and is outside of the scope of this paper.

The company can use the system in the future to manage content for all their social network mobile application. The designs for Buddify CMS were made to be reusable and easily integrated with other application APIs and versions.

While the company is satisfied with the final project result, there were still many aspects that can be optimized. In my opinion, there are always potential for improvements in code structure, functional logic, user experience design and application performance. I will continue to do code refactoring to improve readability, maintainability and reduce complexity.

References

Printed sources

Bass, L., Clements, P. & Kazman, R. 2003. Software Architecture in Practice. 2nd ed. Boston: Addison Wesley.

Crockford, D. 2008. JavaScript: The Good Parts. 1st ed. O'Reilly Media: O'Reilly.

Flanagan, D. 2011. JavaScript: The Definitive Guide. 6th ed. Fonenix: O'Reilly.

Mauthe, A. & Thomas, P. 2005. Professional Content Management Systems: Handling Digital Media Assets. 2nd ed. John Wiley

Electronic sources

Atlassian. No date. Accessed 2018. https://www.atlassian.com/software/jira

Atlassian. No date. Accessed 2018. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Atlassian. No date. What is Git. Accessed 2018.
https://www.atlassian.com/git/tutorials/what-is-git/

Atom. 2018. Accessed 2018. https://atom.io/

Bejan, A. & Lorente, S. 2006. Constructal theory of generation of configuration in nature and engineering. Journal of Applied Physics. Accessed 2018.
https://aip.scitation.org/doi/10.1063/1.2221896

Geary D. 2016. Manage state with Redux-Part 1 Introducing Redux. IBM developerWorks. Accessed 2018. https://www.ibm.com/developerworks/library/wa-manage-state-with-reduxp1-david-geary/index.html

GitHub. 2017. Usage. Webpack. Accessed 2018.
https://github.com/webpack/docs/wiki/usage

Ivanecky, N. 2016. Crash Article in Agile Development. Medium. Accessed 2018.
https://medium.com/open-product-management/crash-article-in-agile-development-da960861259e

JSX. 2014. JSX Specification. Accessed 2018. https://facebook.github.io/jsx/

Lindley, C. 2014. Front-end Driven Applications- A New Approach to Applications. Telerik. Accessed 2018. https://developer.telerik.com/featured/front-end-driven-applications-new-approach-applications/

MDN web docs. No date. Introduction to the DOM. Accessed 2018.
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

OAuth. 2018. OAuth 2.0. Accessed 2018. https://oauth.net/2/

React. No date. Accessed 2018. https://reactjs.org/

React. No date. Introducing JSX. Accessed 2018. https://reactjs.org/docs/introducingjsx.html

React. No date. React-Component. Accessed 2018. https://reactjs.org/docs/react-component.html

Redux. No date. Accessed 2018. https://redux.js.org/

Slack. No date. Accessed 2018. https://slack.com/

Terpil, J. 2016. Redux-From twitter hype to production. Accessed 2018.
http://slides.com/jenyaterpil/redux-from-twitter-hype-to-production#/

Tutorials point. No date. SDLC-Agile Model. Accessed 2018.
https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

Tutorials point. No date. SDLC-Overview. Accessed 2018.
https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

Wilmott, D. 2017. Understanding React’s Virtual DOM vs. the real DOM. Medium. Accessed

2018. https://medium.com/@hidace/understanding-reacts-virtual-dom-vs-the-real-dom68ae29039951

W3C. 2016. Cascading Style Sheets. Accessed 2018. https://www.w3.org/Style/CSS/

W3C. 2017. HTML 5.2. Accessed 2018. https://www.w3.org/TR/html52/

W3chools. No date. JavaScript HTML DOM. Accessed 2018. https://www.w3schools.com/js/js_htmldom.asp

Figures

Figure 1: Client-server model.. 9

Figure 2: The HTML DOM tree of objects (W3schools) 10

Figure 3: Software Development Life Cycle 11

Figure 4: Agile Methodology (Ivanecky 2016) 12

Figure 5: Buddify CMS system model 15

Figure 6: Top level use cases for Buddify Content Management System 17

Figure 7: Process of sending a typical API request through the web service module 19

Figure 8: An example of JSX (React) 20

Figure 9: Comparison between Virtual DOM and Browser DOM in different stages of state change ....

22 Figure 10: State change process of Buddify CMS with Redux (Terpil 2016)

.... 23

Figure 11: Example of Compass folder structure 25

Figure 12: Functionality of Webpack module bundler (GitHub 2017) 26

Figure 13: Atom user interface 27

Figure 14: Representation of Gitflow Workflow branches (Atlassian) 28

Figure 15: Jira Software issue tracker (Atlassian) 29

Figure 16: Slack user interface on desktop and mobile platforms (Slack) 30

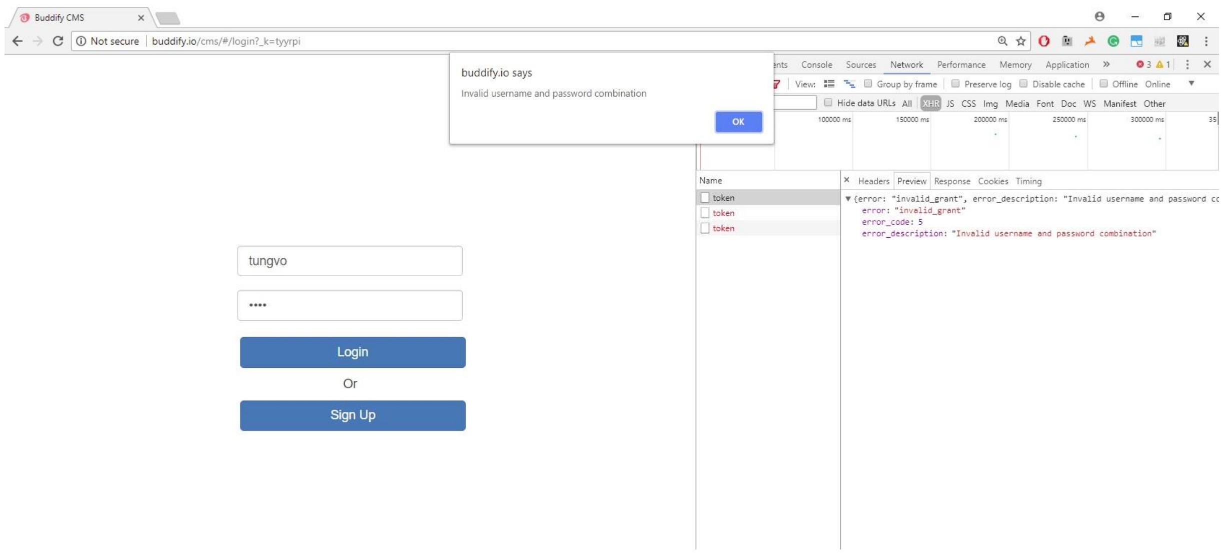
Figure 17: Buddify CMS navigation bar 33

Figure 18: Successful request for flagged users 33

Figure 19: Confirmation popup windows for users 33

Figure 20: Confirmation popup windows for posts and comments 34

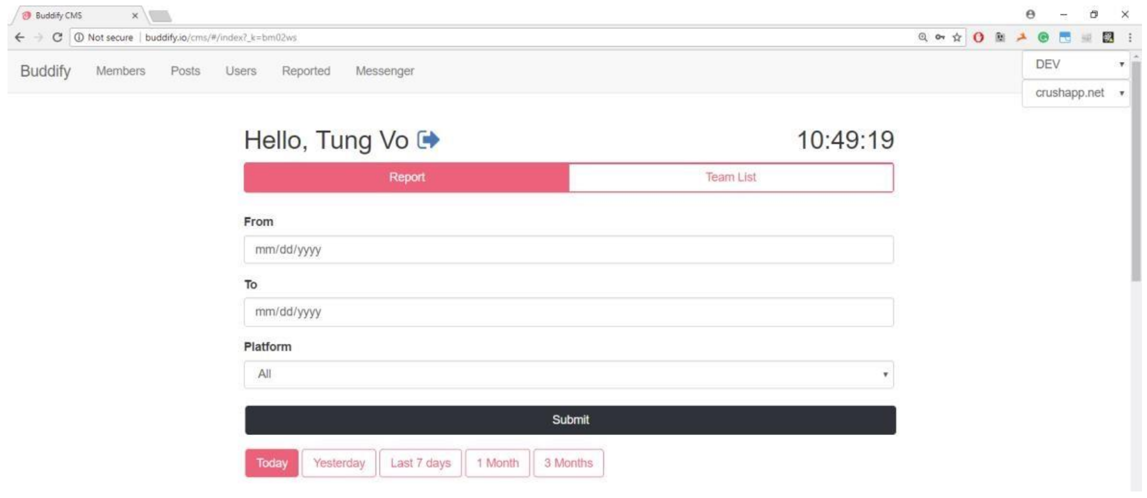
Appendix 1: Authentication service


buddify.io says

Invalid username and password combination

```
▼{error: "invalid_grant", error_description: "Invalid username and password co
    error: "invalid_grant"
    error_code: 5
    error_description: "Invalid username and password combination"
```

Invalid Username password display



Buddify CMS

Not secure | buddify.io/cms/#/index?_k=bm02ws

Buddify Members Posts Users Reported Messenger

DEV

crushapp.net

Hello, Tung Vo 10:49:19

Report | Team List

From

mm/dd/yyyy

To

mm/dd/yyyy

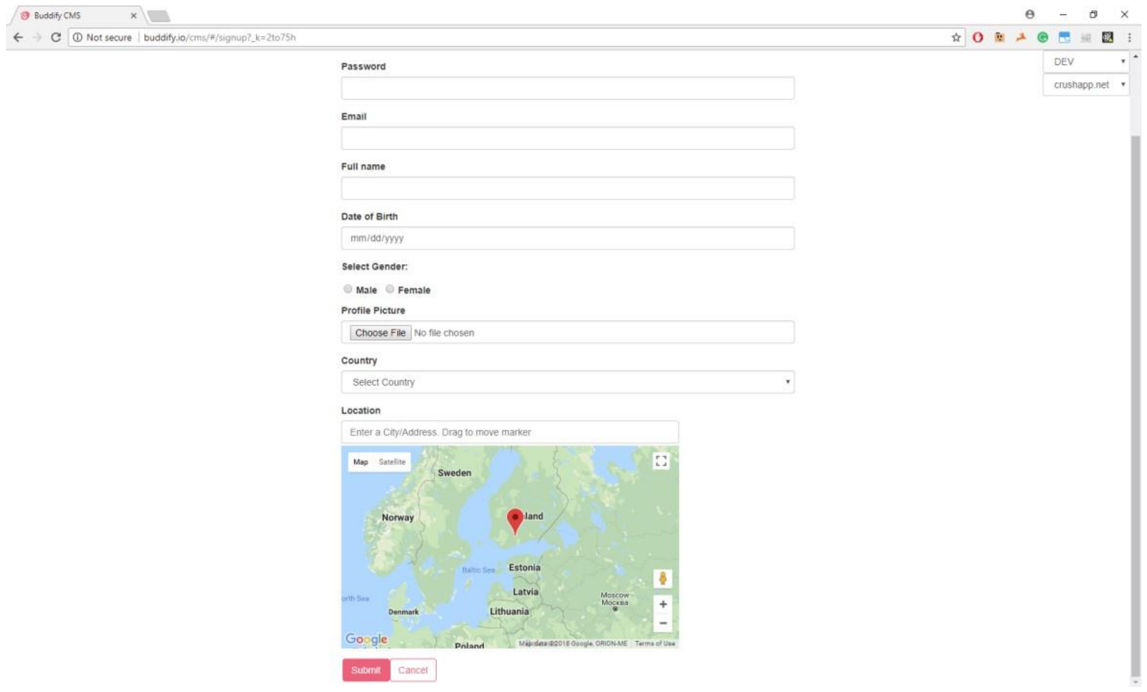
Platform

All

Submit

Today | Yesterday | Last 7 days | 1 Month | 3 Months

Landing page on successful login.



Buddify CMS

Not secure | buddify.io/cms/#/signup?_k=2to75h

DEV

crushapp.net

Password

Email

Full name

Date of Birth

mm/dd/yyyy

Select Gender:

Male Female

Profile Picture

Choose File No file chosen

Country

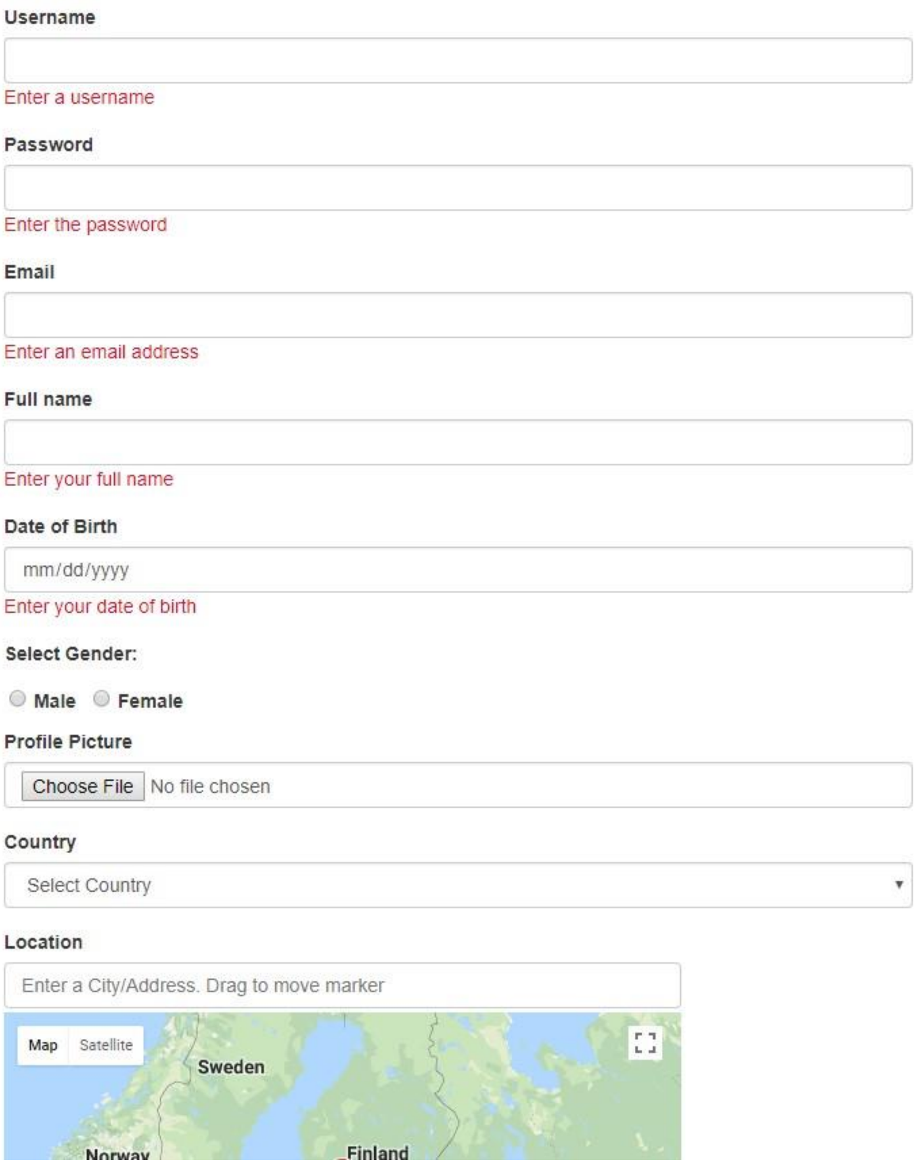
Select Country

Location

Enter a City/Address. Drag to move marker

Submit Cancel

Sign up page.



Username

Enter a username

Password

Enter the password

Email

Enter an email address

Full name

Enter your full name

Date of Birth

mm/dd/yyyy

Enter your date of birth

Select Gender:

Male Female

Profile Picture

Choose File No file chosen

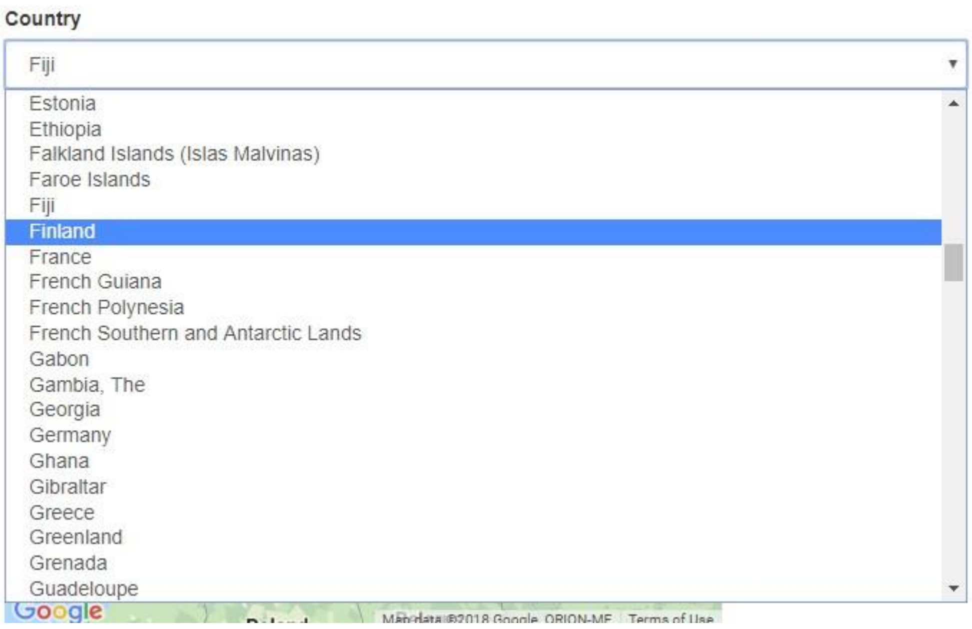
Country

Select Country

Location

Enter a City/Address. Drag to move marker

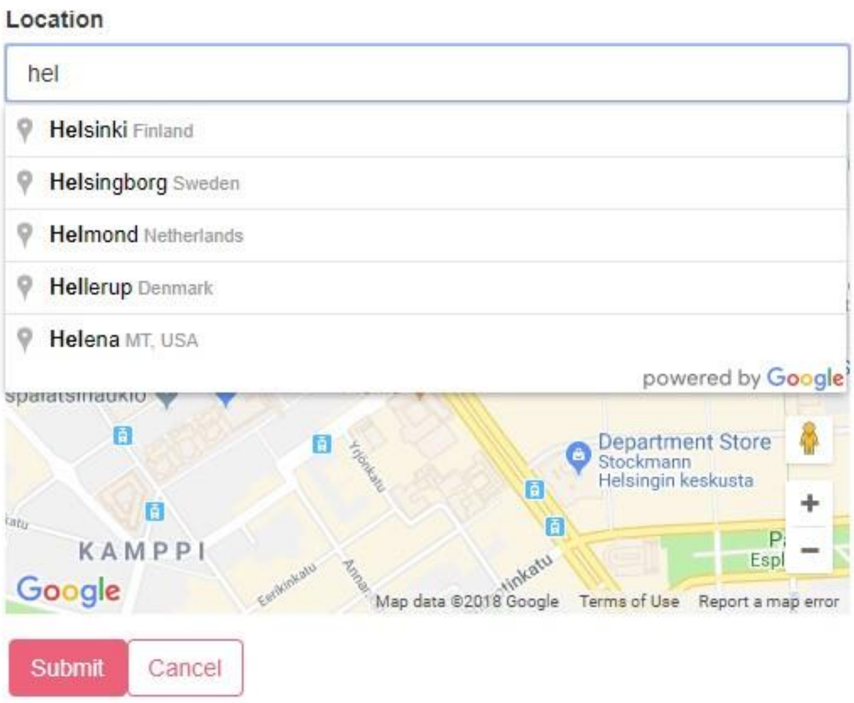
Required Signup fields.


Country

Fiji

Estonia
Ethiopia
Falkland Islands (Islas Malvinas)
Faroe Islands
Fiji
Finland
France
French Guiana
French Polynesia
French Southern and Antarctic Lands
Gabon
Gambia, The
Georgia
Germany
Ghana
Gibraltar
Greece
Greenland
Grenada
Guadeloupe

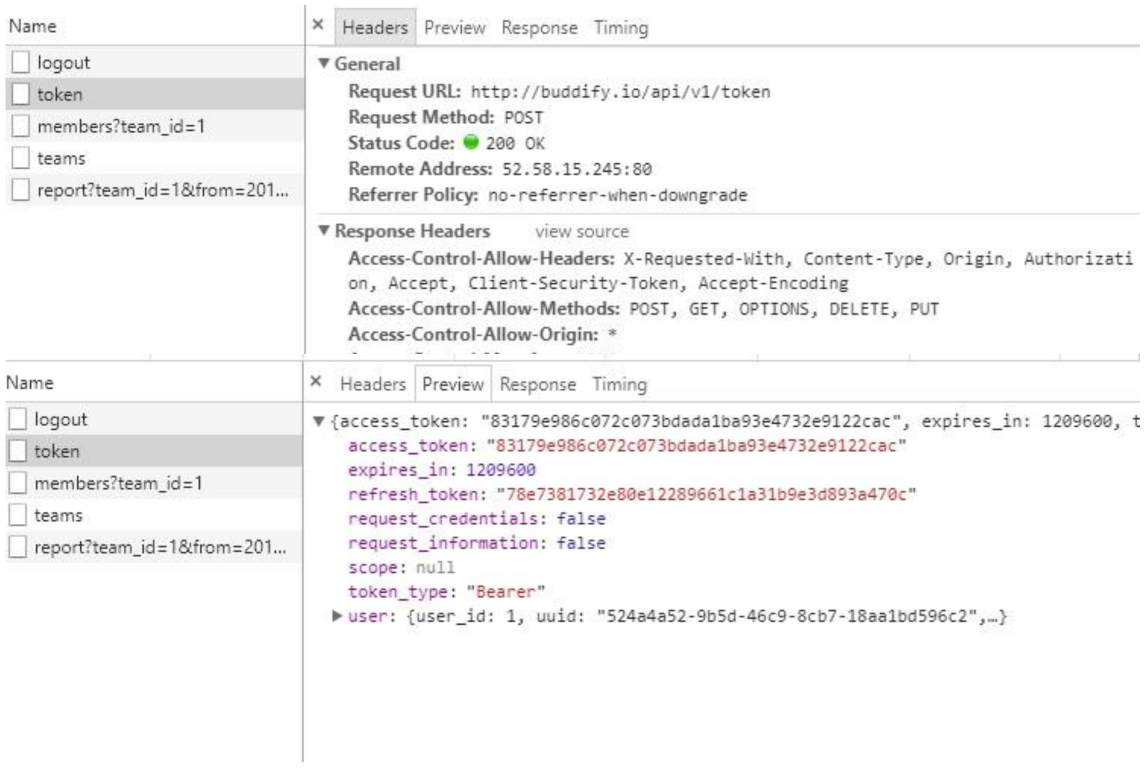
Country drop-down options.


Location

hel

Helsinki Finland
Helsingborg Sweden
Helmond Netherlands
Hellerup Denmark
Helena MT, USA

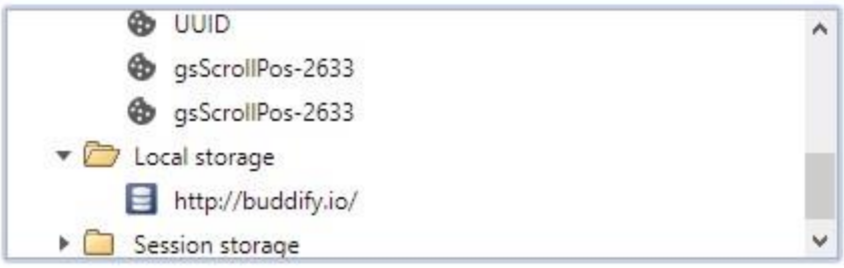
Location auto complete feature.

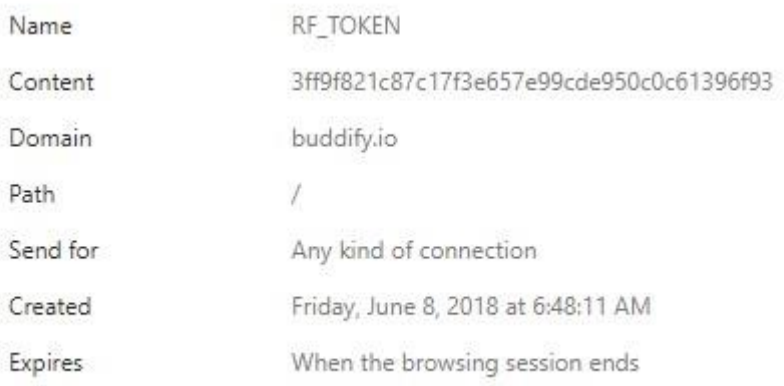


```
▼ General
    Request URL: http://buddify.io/api/v1/token
    Request Method: POST
    Status Code: ● 200 OK
    Remote Address: 52.58.15.245:80
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Access-Control-Allow-Headers: X-Requested-With, Content-Type, Origin, Authorizati
    on, Accept, Client-Security-Token, Accept-Encoding
    Access-Control-Allow-Methods: POST, GET, OPTIONS, DELETE, PUT
    Access-Control-Allow-Origin: *
```

```
▼ {access_token: "83179e986c072c073bdada1ba93e4732e9122cac", expires_in: 1209600, t
    access_token: "83179e986c072c073bdada1ba93e4732e9122cac"
    expires_in: 1209600
    refresh_token: "78e7381732e80e12289661c1a31b9e3d893a470c"
    request_credentials: false
    request_information: false
    scope: null
    token_type: "Bearer"
  ▶ user: {user_id: 1, uuid: "524a4a52-9b5d-46c9-8cb7-18aa1bd596c2",…}
```

Access token and refresh token are provided on successful login.

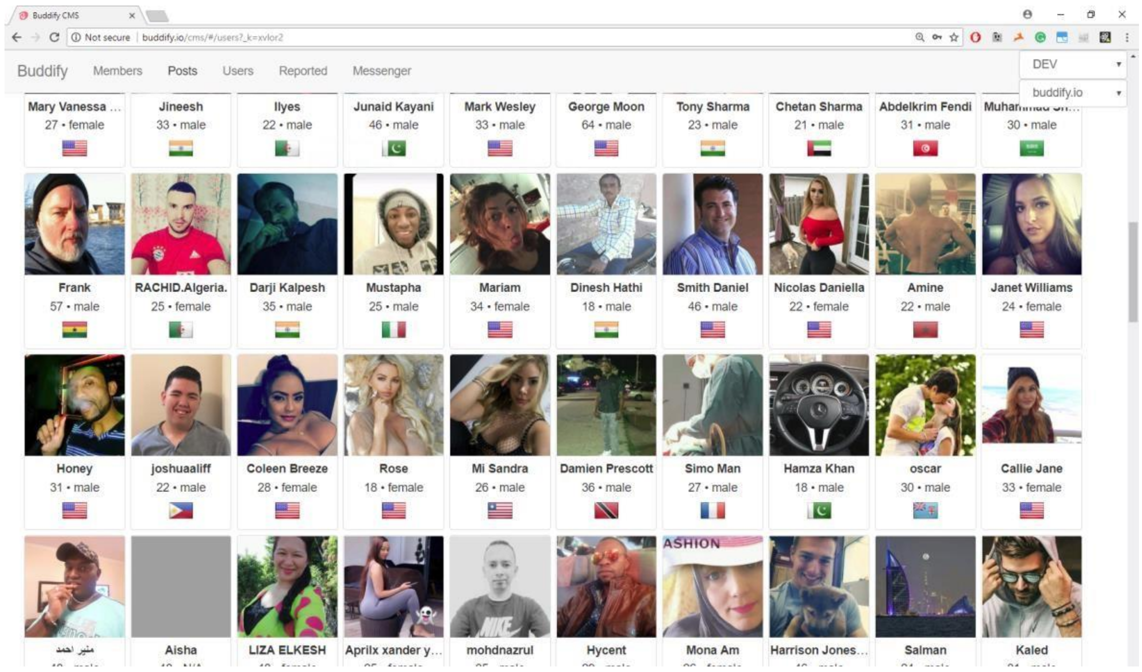




Name	RF_TOKEN
Content	3ff9f821c87c17f3e657e99cde950c0c61396f93
Domain	buddify.io
Path	/
Send for	Any kind of connection
Created	Friday, June 8, 2018 at 6:48:11 AM
Expires	When the browsing session ends

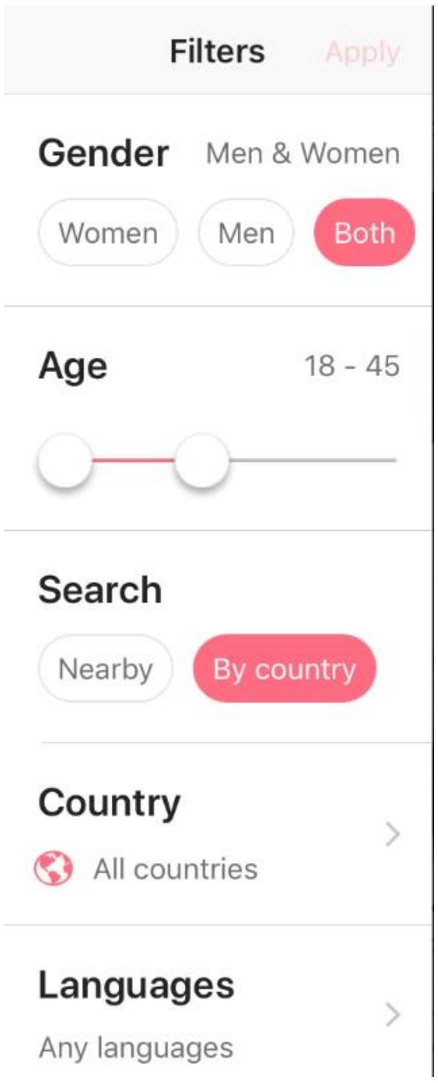
Tokens are stored on client side.

Appendix 2: User section



Mary Vanessa ... 27 • female	Jineesh 33 • male	Ilyes 22 • male	Junaid Kayani 46 • male	Mark Wesley 33 • male	George Moon 64 • male	Tony Sharma 23 • male	Chetan Sharma 21 • male	Abdelkrim Fendi 31 • male	Muhan 30 • male
Frank 57 • male	RACHID.Algeria. 25 • female	Darji Kalpesh 35 • male	Mustapha 25 • male	Mariam 34 • female	Dinesh Hathi 18 • male	Smith Daniel 46 • male	Nicolas Daniella 22 • female	Amine 22 • male	Janet Williams 24 • female
Honey 31 • male	joshuaaliff 22 • male	Coleen Breeze 28 • female	Rose 18 • female	Mi Sandra 26 • male	Damien Prescott 36 • male	Simo Man 27 • male	Hamza Khan 18 • male	oscar 30 • male	Callie Jane 33 • female
منير احمد	Aisha	LIZA ELKESH	Aprilx xander y...	mohdnazrul	Hycent	Mona Am	Harrison Jones...	Salman	Kaled

User exploration based on last seen online order.


Filters
Apply

Gender: Men & Women
Women | Men | [✓ Both]

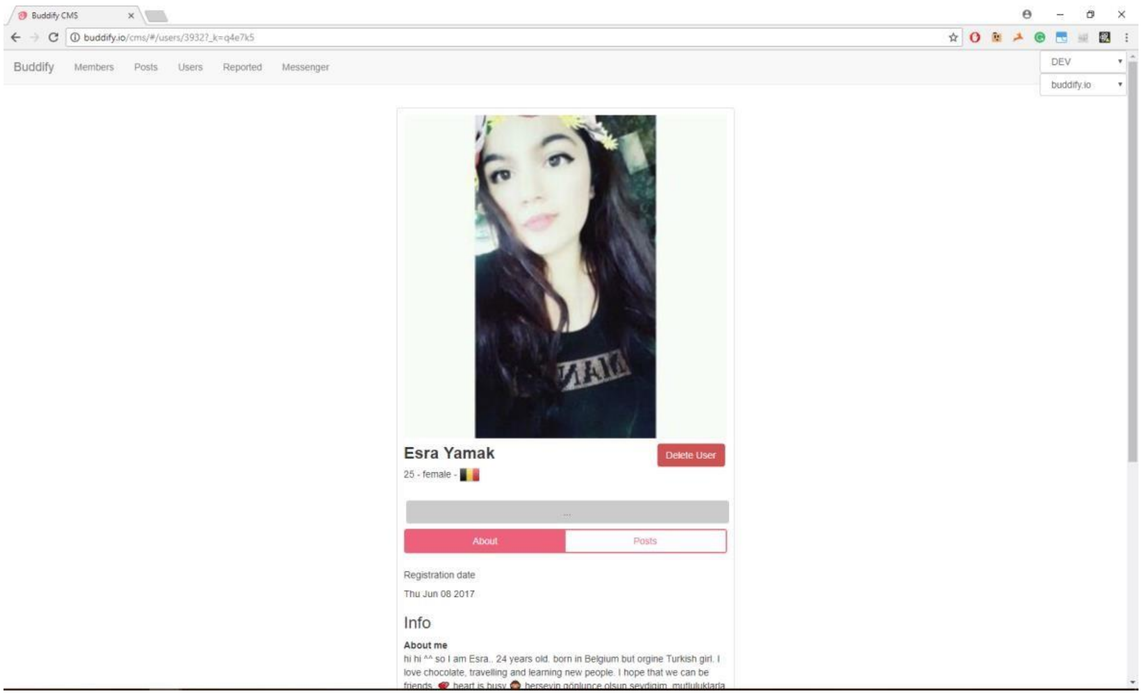
Age: 18 - 45

Search
Nearby | [✓ By country]

Country
All countries

Languages
Any languages

Filters for users.



Esra Yamak

25 - female -

Registration date

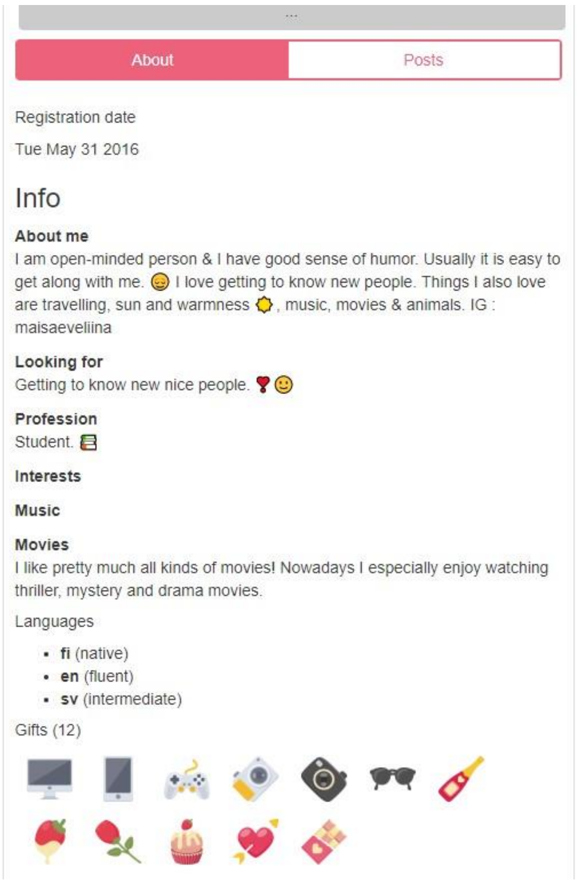
Thu Jun 08 2017

Info

About me

hi hi ^^ so I am Esra.. 24 years old. born in Belgium but orgine Turkish girl. I love chocolate, travelling and learning new people. I hope that we can be friends. heart is busy hersevin gönlunce olsun sevdigim. mutluluklarla

User information page.



Registration date

Tue May 31 2016

Info

About me

I am open-minded person & I have good sense of humor. Usually it is easy to get along with me. I love getting to know new people. Things I also love are travelling, sun and warmness , music, movies & animals. IG : maisaeveliina

Looking for

Getting to know new nice people.

Profession

Student.

Interests

Music

Movies

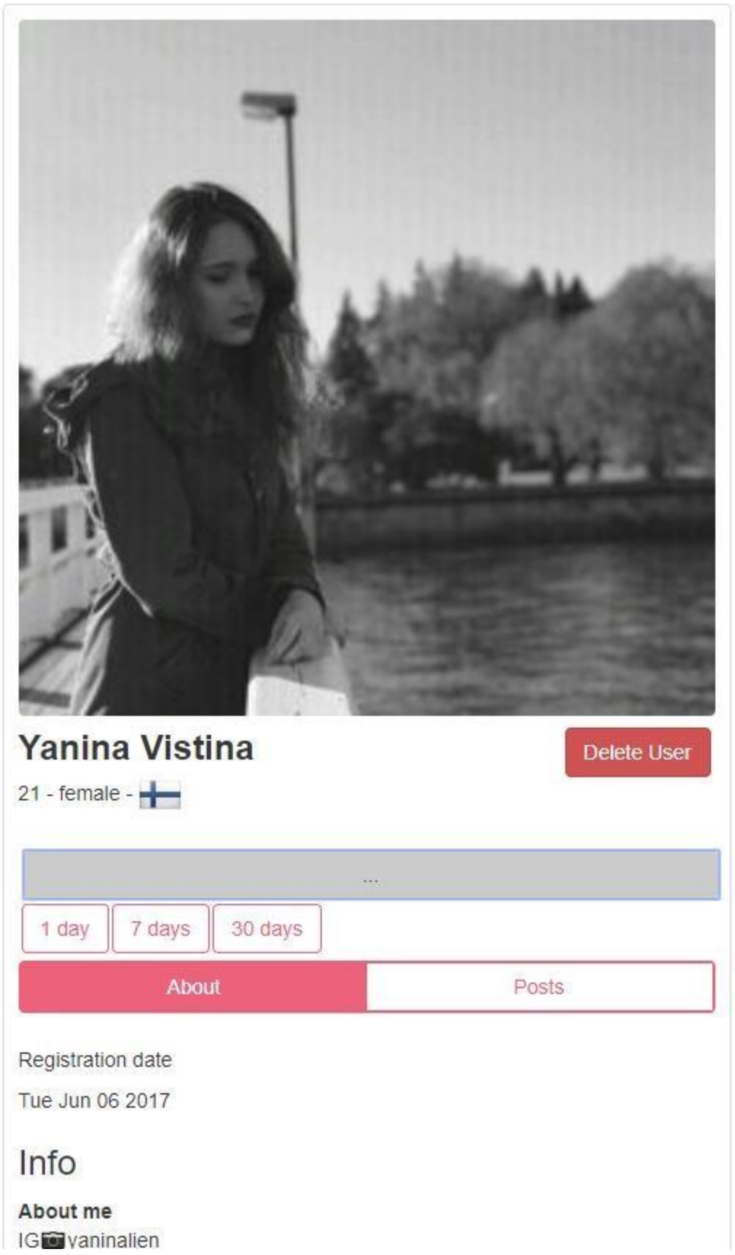
I like pretty much all kinds of movies! Nowadays I especially enjoy watching thriller, mystery and drama movies.

Languages

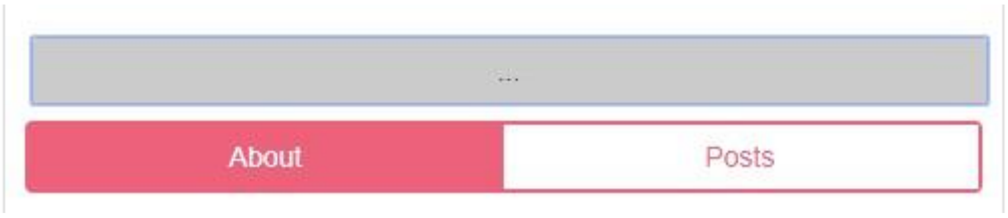
- **fi** (native)
- **en** (fluent)
- **sv** (intermediate)

Gifts (12)

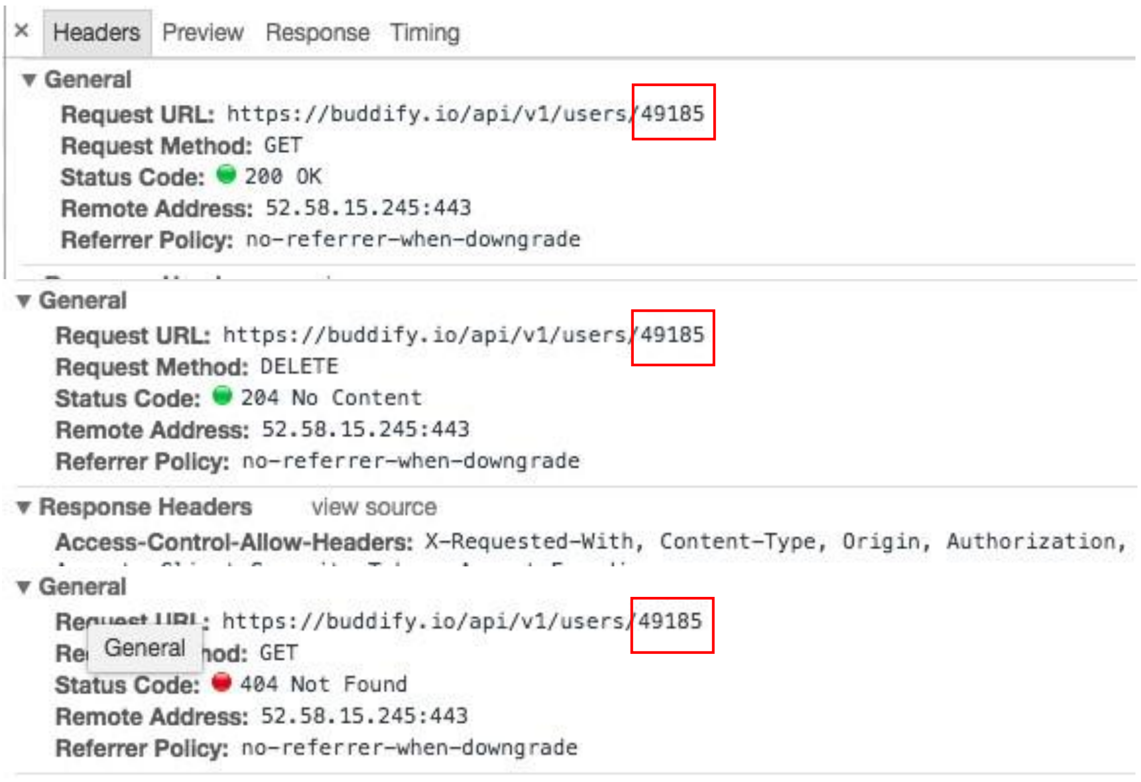
All information category is displayed according to the Buddify app.



User delete and ban functions.

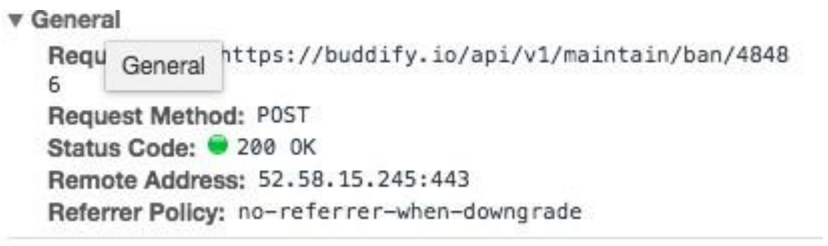


Ban functions are hidden within the button.



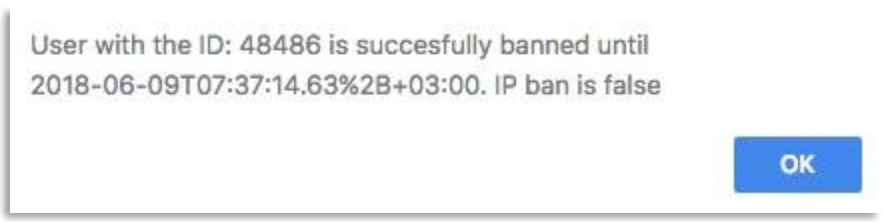
```
× Headers  Preview  Response  Timing
▼ General
    Request URL: https://buddify.io/api/v1/users/49185
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 52.58.15.245:443
    Referrer Policy: no-referrer-when-downgrade

▼ General
    Request URL: https://buddify.io/api/v1/users/49185
    Request Method: DELETE
    Status Code: ● 204 No Content
    Remote Address: 52.58.15.245:443
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Access-Control-Allow-Headers: X-Requested-With, Content-Type, Origin, Authorization,

▼ General
    Request URL: https://buddify.io/api/v1/users/49185
    Re General hod: GET
    Status Code: ● 404 Not Found
    Remote Address: 52.58.15.245:443
    Referrer Policy: no-referrer-when-downgrade
```

Delete feature is functional and behaving as intended. Once a user is delete, the user profile can no longer be retrieved.



```
▼ General
    Requ General https://buddify.io/api/v1/maintain/ban/4848
    6
    Request Method: POST
    Status Code: ● 200 OK
    Remote Address: 52.58.15.245:443
    Referrer Policy: no-referrer-when-downgrade
```

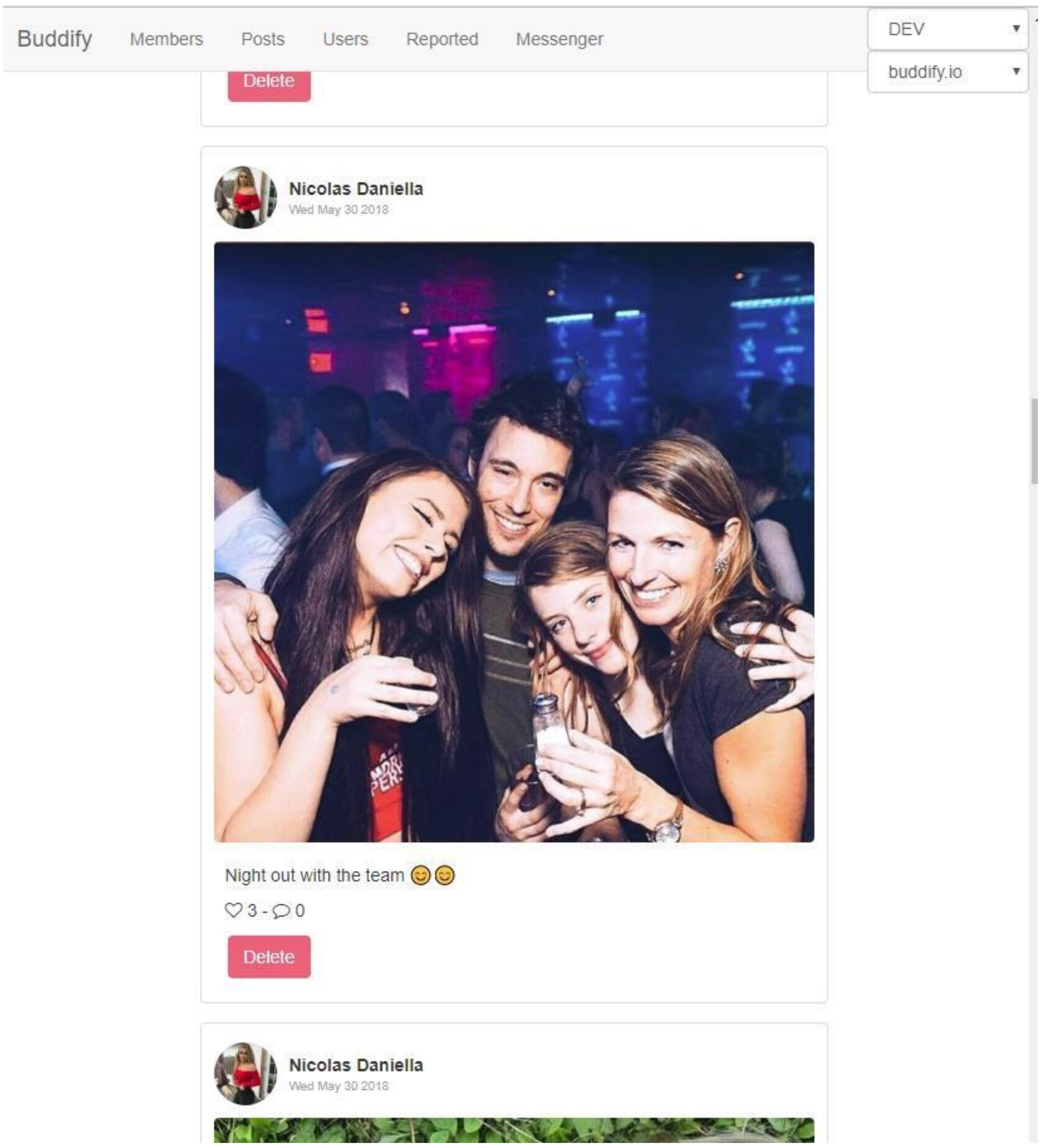
Ban feature is functional and behaving as intended.



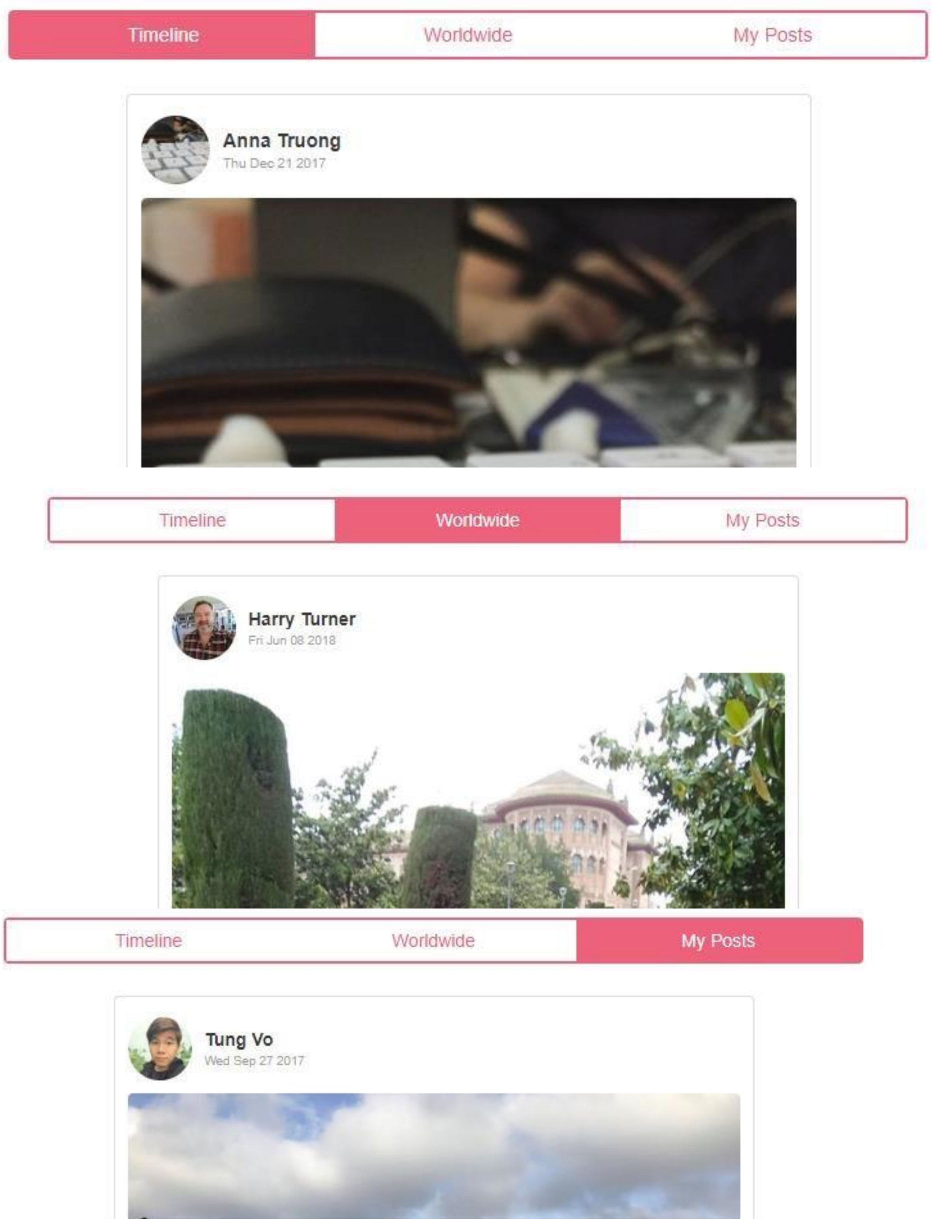
User with the ID: 48486 is succesfully banned until 2018-06-09T07:37:14.63%2B+03:00. IP ban is false

Popup when a user is successfully banned.

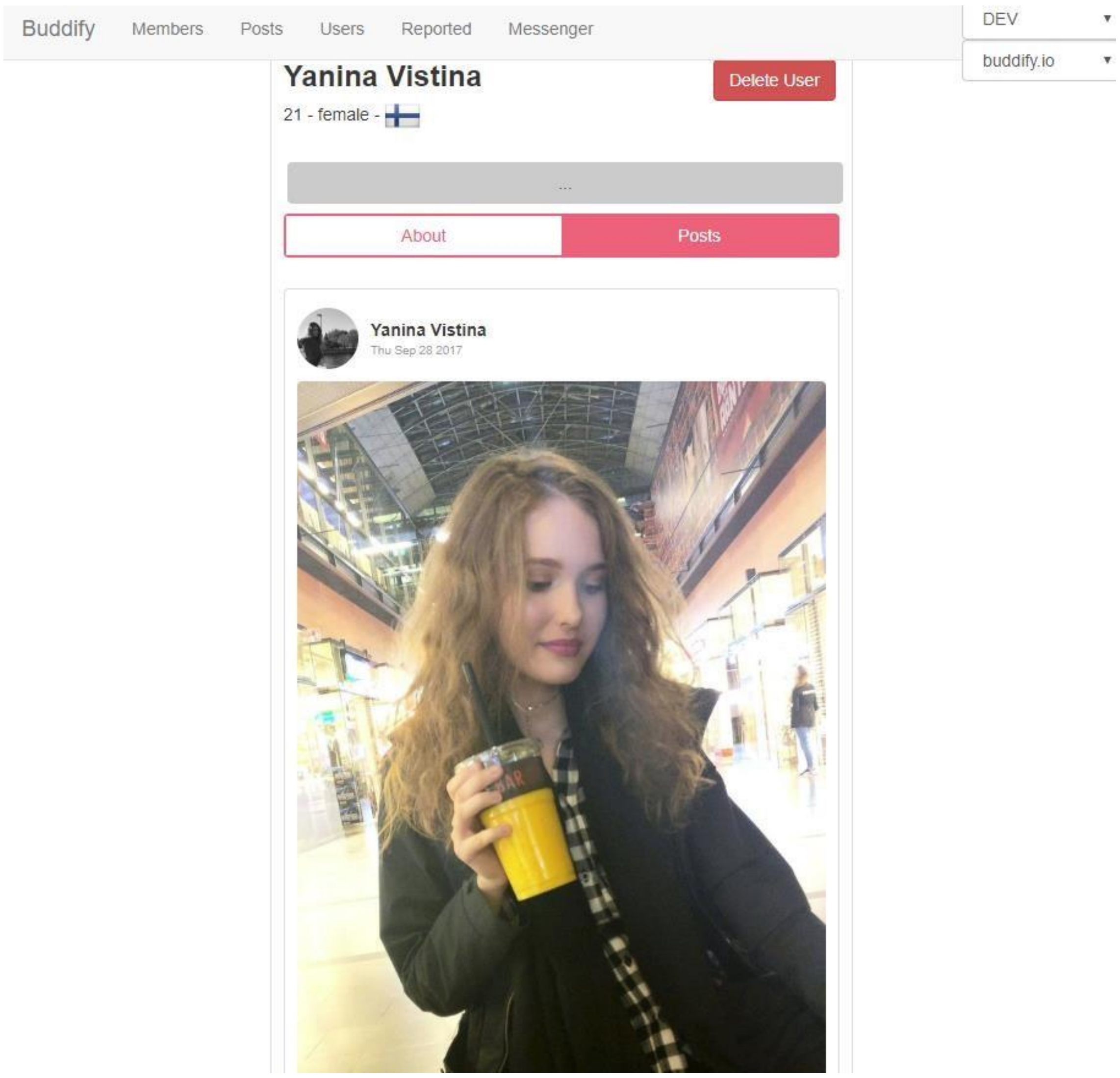
Appendix 3: Posts and Comments section



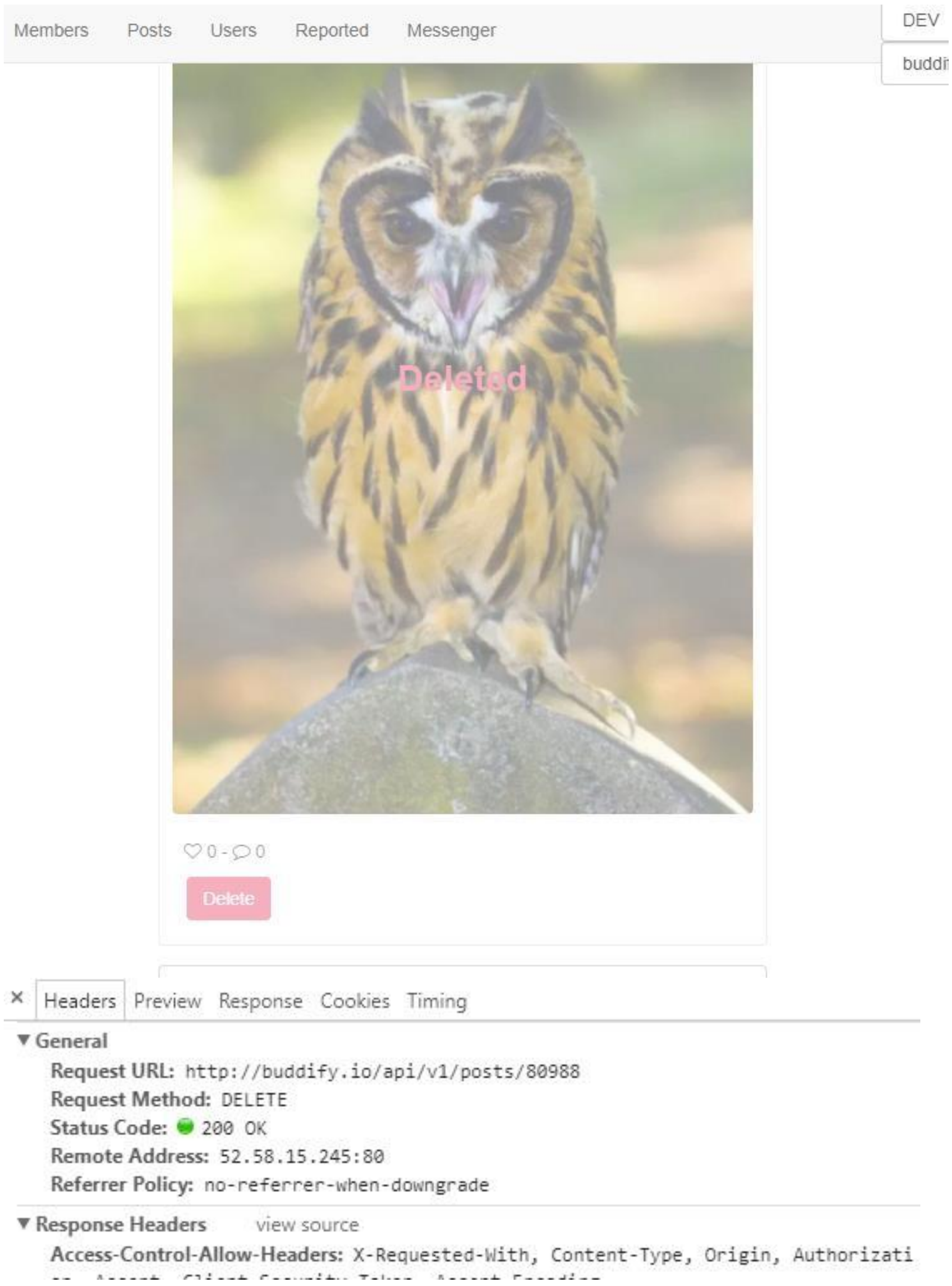
Post section UI.



Three different feeds provided.



Posts inside the User profile section.

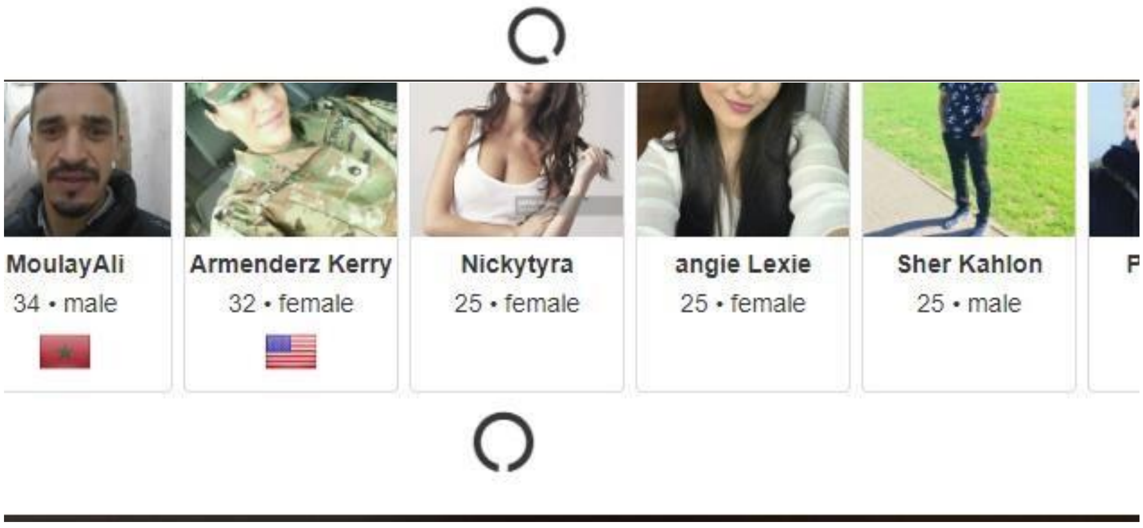


```
▼ General
    Request URL: http://buddify.io/api/v1/posts/80988
    Request Method: DELETE
    Status Code: 200 OK
    Remote Address: 52.58.15.245:80
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Access-Control-Allow-Headers: X-Requested-With, Content-Type, Origin, Authorizati
```

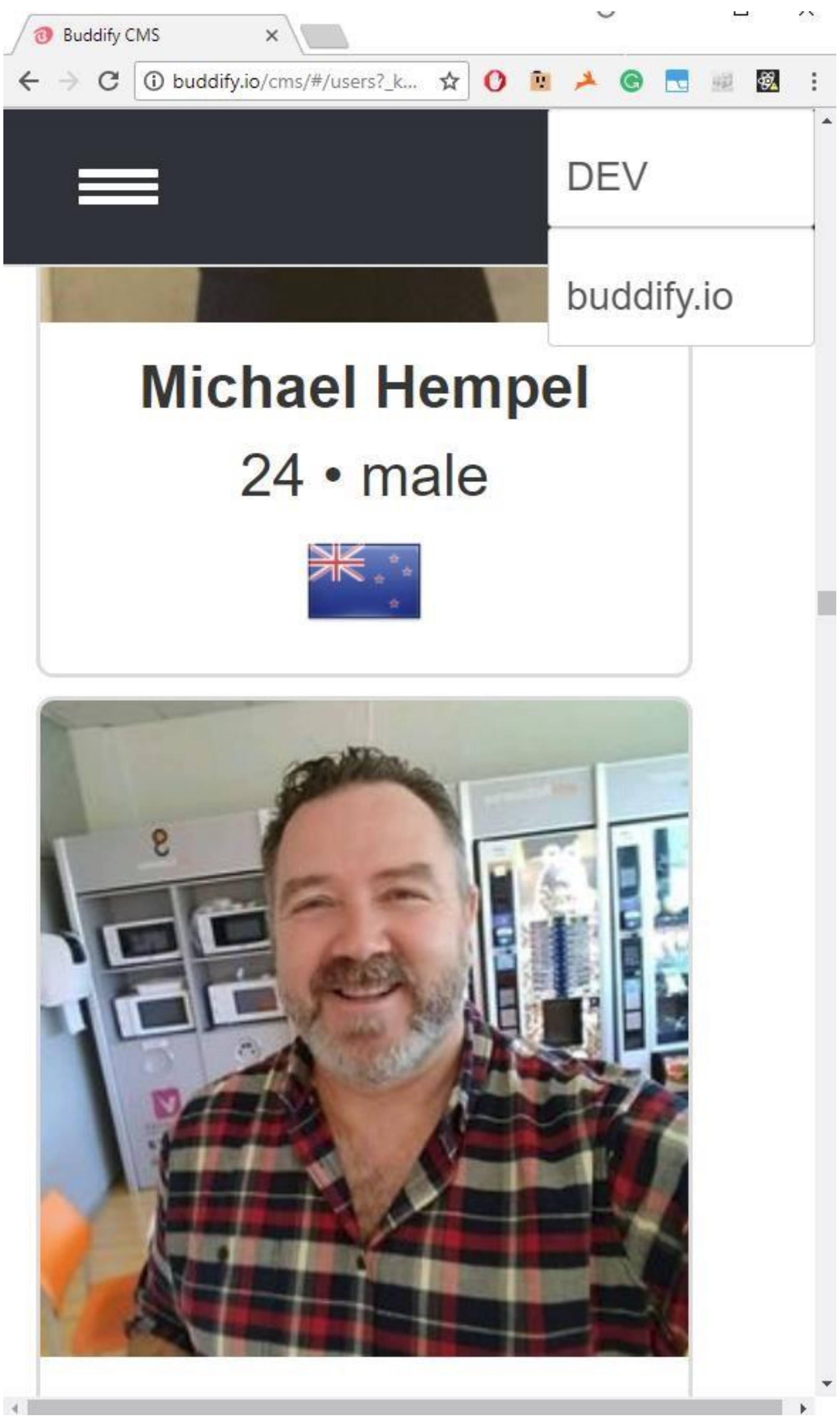
Delete feature is functional and behaving as intended.
Appendix 4: Quality requirements

Tbt to my last vietnam trip ❤ I dont know who was more shy in this photo moment, the little girl or me 😅

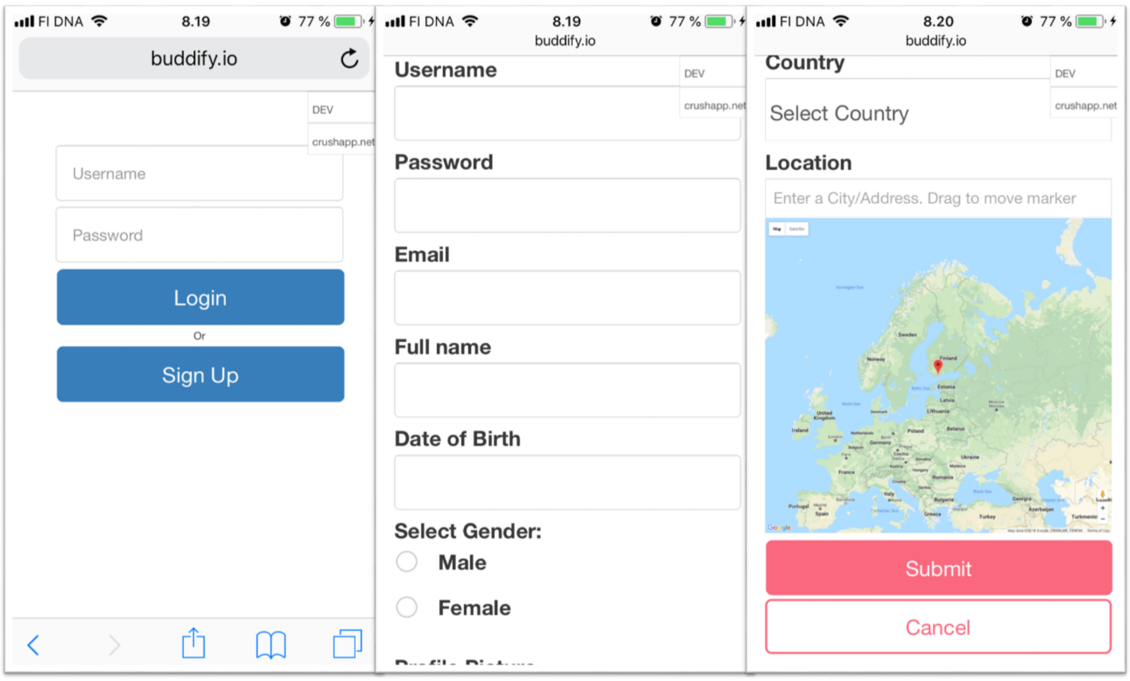
♡ 17 - 💬 11



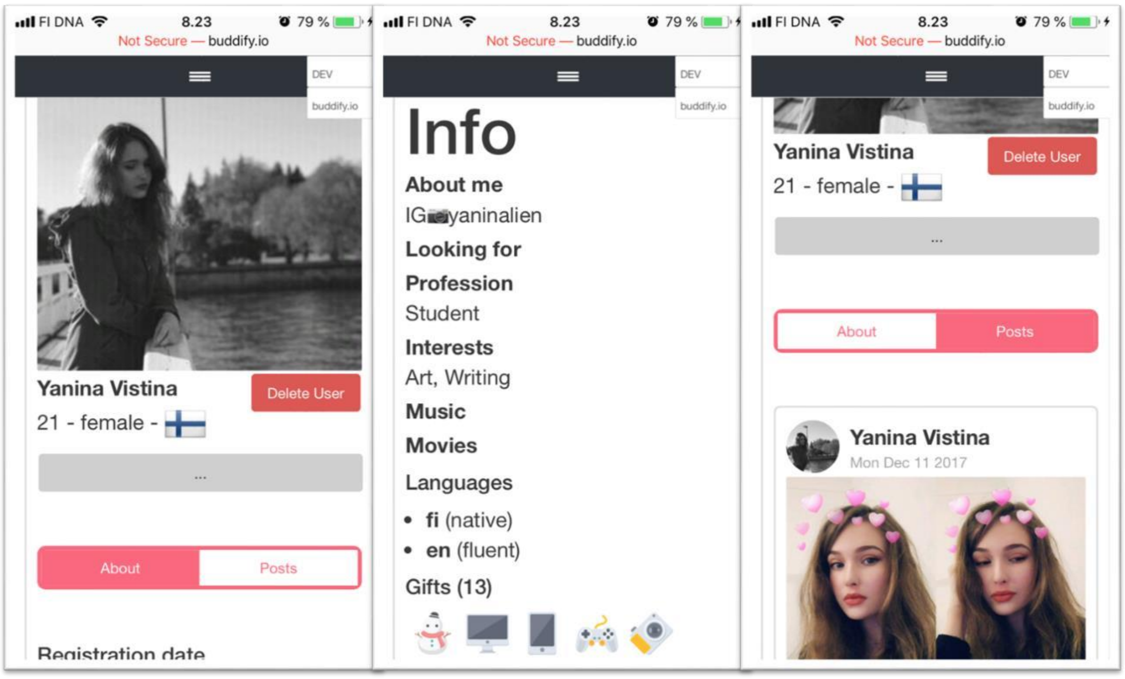
Loading indicators



Responsive UI on website



Login and signup section on mobile view.



Yanina Vistina
21 - female -

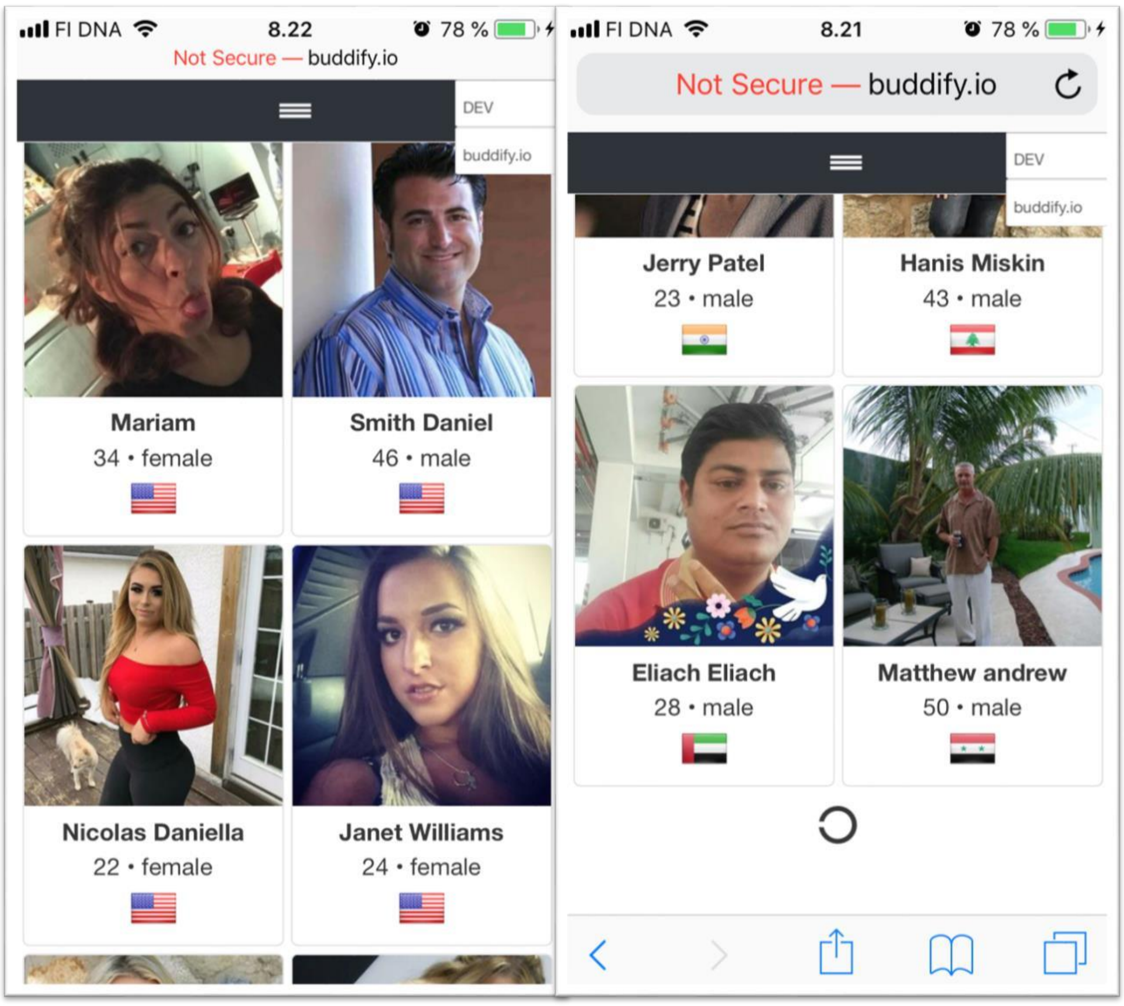
Info

About me
IG📷yaninalien
Looking for
Profession
Student
Interests
Art, Writing
Music
Movies
Languages
- fi (native)
- en (fluent)

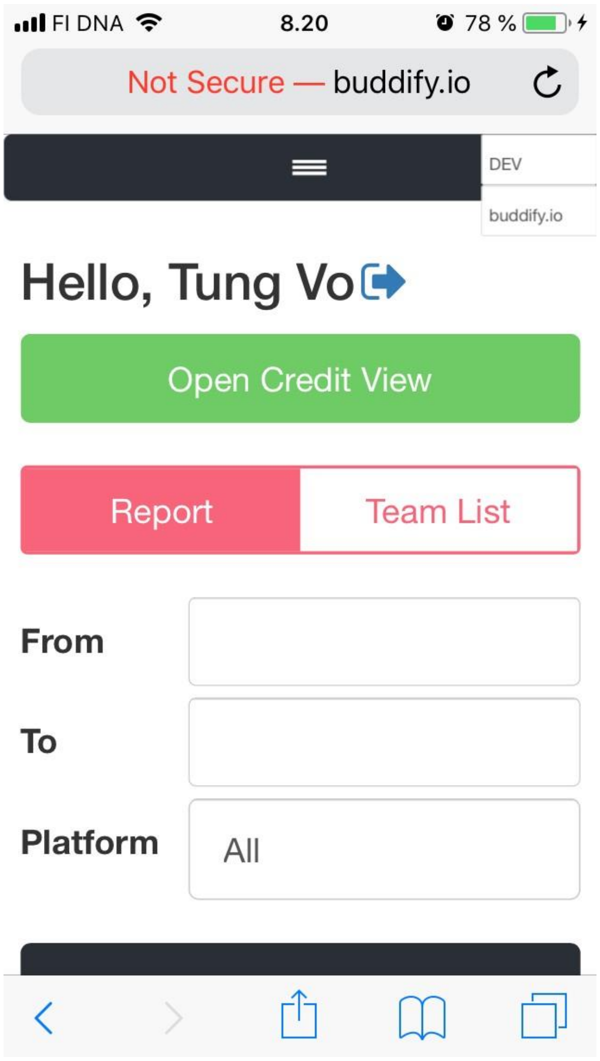
Gifts (13)

Yanina Vistina
Mon Dec 11 2017

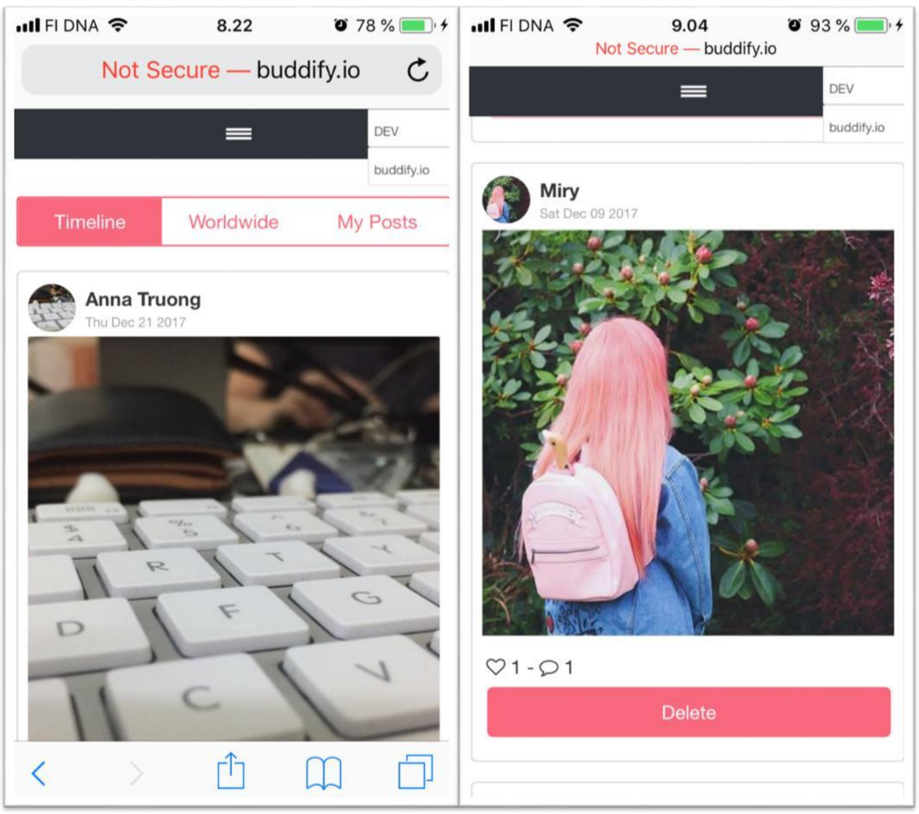
User profile on mobile view.



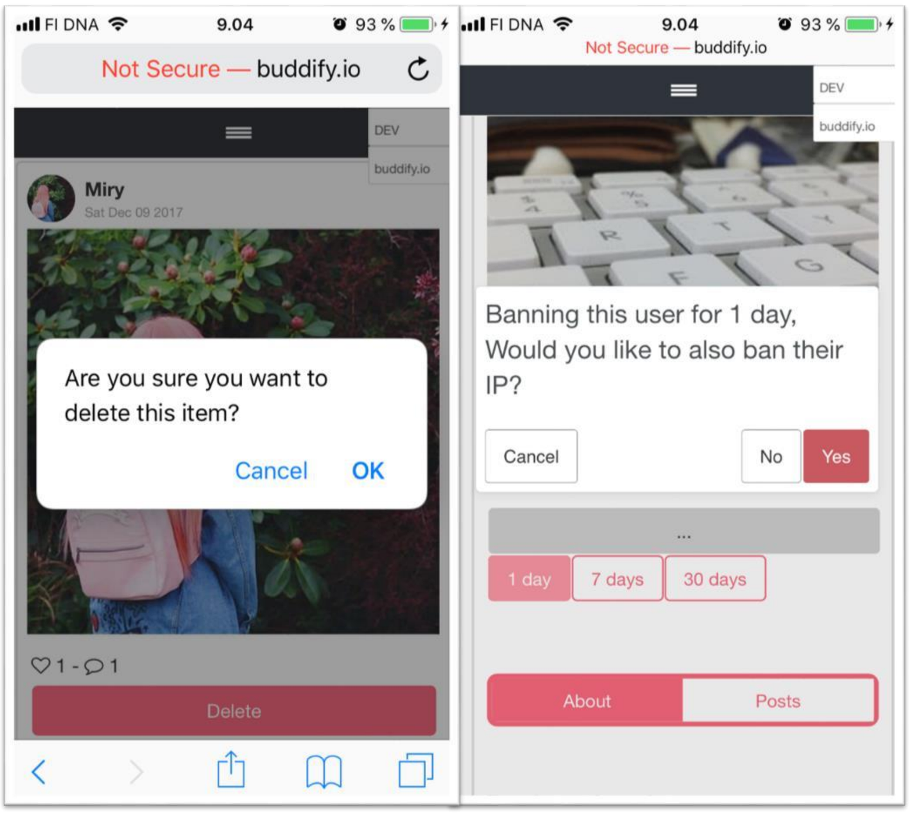
User Page on mobile view.



Landing page on mobile view.



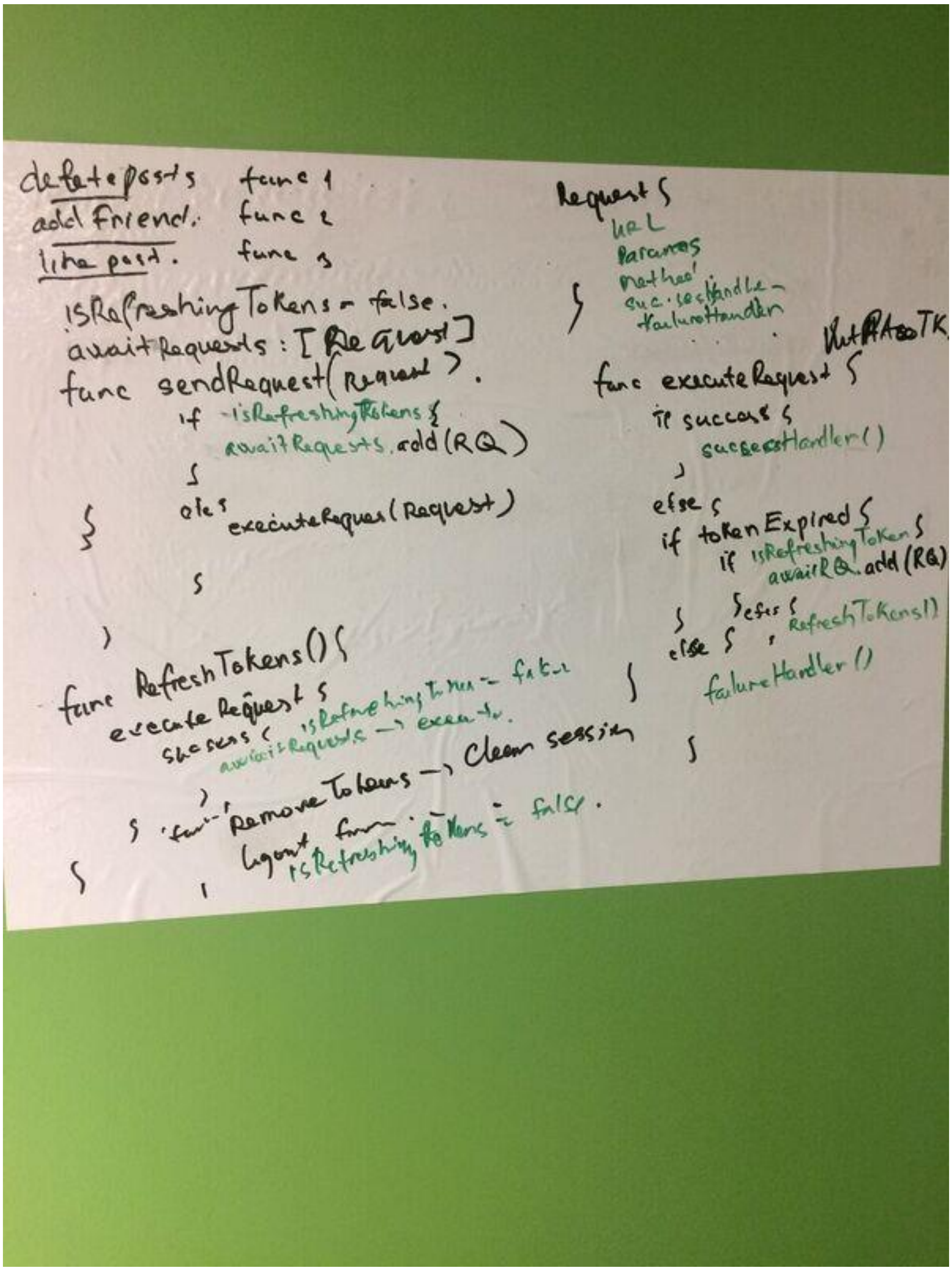
Posts section on mobile view


Are you sure you want to delete this item?

Banning this user for 1 day, Would you like to also ban their IP?

Popups on mobile view.

Appendix 5: Buddify design plans

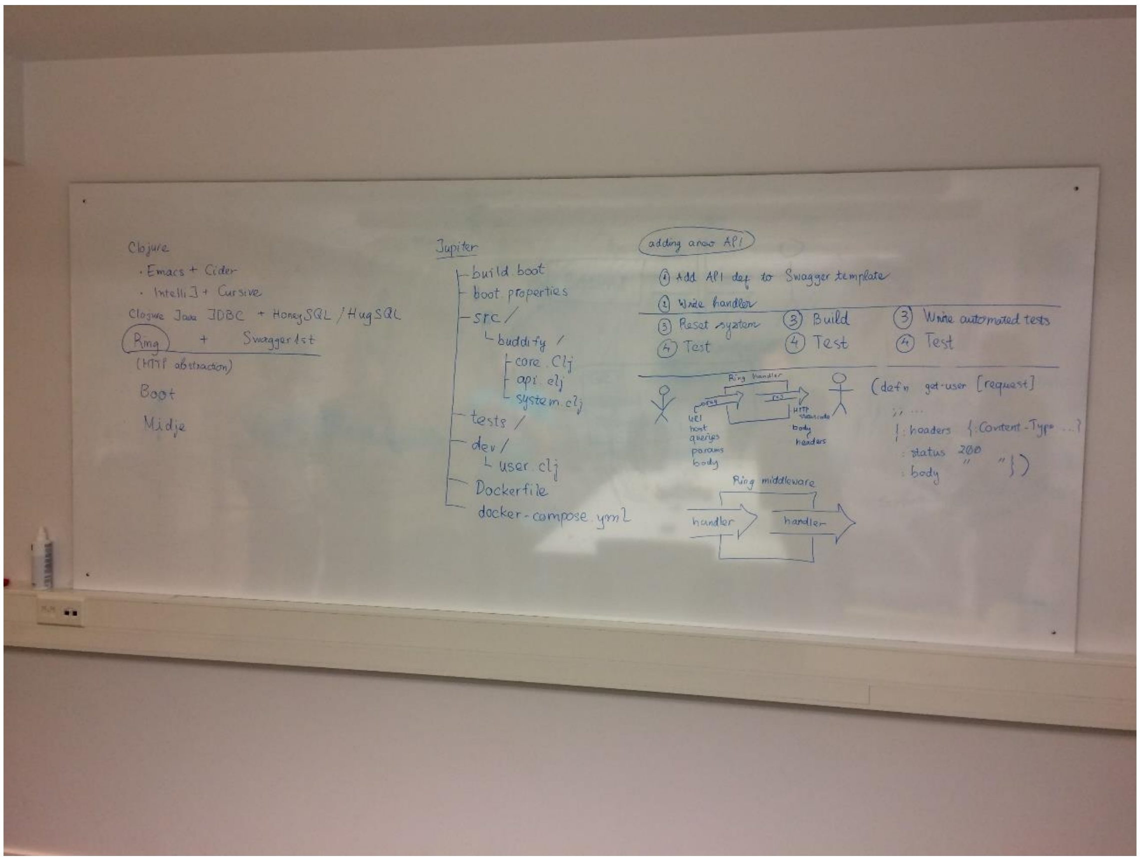


```
deletepasts     func 1
add Friend.     func 2
like post.      func 3

  isRefreshingTokens = false.
  awaitRequests : [Request]
  func sendRequest(Request).
          if isRefreshingTokens {
              awaitRequests.add(RQ)
          }
  }       else executeRequest(Request)

          }
     }

func RefreshTokens() {
    executeRequest {
        success {  isRefreshingTokens = false
                   awaitRequests -> execute.
        }
   } fail RemoveTokens -> Clean session
           Logout from
           isRefreshingTokens = false.
}
```

```
Request {
       URL
       Params
       method
}      successHandler
       failureHandler
                          WithAPPTK

func executeRequest {
     if success {
          successHandler()
     }
     else {
     if tokenExpired {
         if isRefreshingToken {
             awaitRQ.add(RQ)
         } else { RefreshTokens()
     } else {
     failureHandler()
  }                                     }
     }
```

Authentication service design.

Buddify initial project plans.



Buddify backend architecture designs.