THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

INVESTIGATING OPEN BADGES, WHAT THEY ARE & HOW THEY ARE DISPLAYED

Craig Knott (42336866)

Faculty of ITEE - BE (Mechatronics) & BIT

ABSTRACT

There are three primary components of an Open Badge, issuing, earning and displaying. However to issue an Open Badge requires it first be created, this report investigates the methods of creating and issuing Open Badges, earning badges and the Mozilla Backpack used for managing badges earned from a number of issuers, and displaying badges using both current methods involving the Mozilla Backpack and new displays using the Unity3D game engine.

Badges are created using either signed or hosted badge assertions. An assertion is a JSON object representing a badge; it contains information pertaining to the badge name, description, issuer organisation, and earner identity (typically an email). Once a badge assertion has been created and hosted at a public URL the issuer uses the Backpack API to offer the badge to the recipient.

The best method found for creating and issuing badges is to use the Mozilla BadgeKit and API. This provides a RESTful API for interacting with a badge database. The BadgeKit provides the user with a simple front-end interface to design, create and issue badges. Alternatively the issuing organisation can design their own front end website or integrate the BadgeKit API into their existing website to simplify the creation and issuing of badges.

The Backpack is used to manage earned badges, place them into groups and control the access badge displayers using the Backpack Displayer API have to badge groups. Backpack badge groups can be shared across social media and websites with its built-in sharing functions.

By implementing the BadgeKit and BadgeKit-API a set of sample badges was generated and issued to dummy Backpack accounts for the purpose of demonstrating new badge displayer software. A Unity3D asset package was created to simplify the creation of Open Badge related visualisations in the Unity game engine. It allows any developer to import badges into their scene with simple click and drag. It implements the Backpack displayer API to pull all public badges for a user, requiring only an email address. Finally this asset package was used to create a demonstration visualising badges in the Learning Innovation Building. It was extended to include simple search functionality to let users quickly search for a particular skill and find where in the building someone with a badge matching that skill can be found.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

NOMENCLATURE

Achievements	Awards given in games for completing some challenge tasks, achievements typically have no effect on gameplay.
API	Application Programmer Interface
Assertion	Refers to an Open Badges badge assertion, a set of JSON objects containing information about a badge to be awarded.
Backpack	Refers to the Mozilla Backpack, a web application designed to store badges earned from any number of issuers.
Badge	A digital or physical representation of some achievement.
BadgeKit	Refers to the Mozilla Open Badges BadgeKit, a web application designed to simplify the process of creating and issuing badges.
Express.JS	A minimal and flexible web application framework for Node.JS
HTTP	Hypertext Transfer Protocol
Issuer	Refers to the issuing organisation, the organisation awarding or issuing badges to students or badge earners.
JavaScript	A dynamic computer programming language most commonly used in web browsers.
JSON	JavaScript Object Notation.
JWS	JavaScript Web Signature.
JWT	JSON Web Token, a compact URL-safe means of representing claims to be transferred between two parties.
Levelling	The process of gaining experience and advancing through many types of games.
LMS	Learning Management System.
LTI	Learning Tools Interoperability.

Mozilla Foundation	An open source not for profit organisation and the creators of open badges among many other products.
MVC	Model-View-Controller
MySQL	A popular open source relational database management system.
Node.JS	A JavaScript runtime environment used for building fast scalable network applications.
Open Badges	The Mozilla Open Badges system for creating and awarding digital badges.
PNG	Portable Network Graphics image file.
Prefab	Refers to a set of linked unity game objects setup to easily create multiple instances of the object.
Scout Craft	Covers a variety of knowledge and skills required by scouts.
Script	A small programming script used in the unity game engine to create and manage various in game events.
Texture	An image to be wrapped around an object to make it appear more realistic and give it depth.
Unity3D	A modern game development and game engine.
URL	A web address.

1.0 INTRODUCTION

Education today is completely different from what it used to be one or two decades ago. With the development of the Internet comes a wealth of information accessible to nearly anyone, as long as they have a computer and an Internet connection. Traditional methods of teaching are no longer the rule and many universities and other learning institutions now offer courses run entirely online in addition to their other on campus curriculum.

These online courses offer a great amount of education, however they offer little to no recognition of the skills and knowledge gained throughout the course. Learning is also frequently occurring outside of formal education in areas such as scouting groups or clubs and societies. (eg: chess club).

A major problem lies in bridging the gab between the formal and informal education. A key component of this would involve a way to be able to receive formal recognition for skills and knowledge learned outside of school, university or work. Being able to show this recognition off to prospective employers, friends or colleagues.

This document shall guide you through the process of discovering what a badge is, the difference between digital badges and Open Badges, how Open Badges are created, issued, earned and displayed and how they can be beneficial to society. It will also discuss the potential to create useful and unique displays using badges.

1.1 WHAT IS A BADGE

"badge [baj] *(noun)*: a special or distinctive mark, token, or device worn as a sign of allegiance, membership, authority, achievement, etc." (Dictionary.com, 2014)

An Open Badge is simply a digital version of a badge, this means that there is no physical mark, token or device to be worn and that the sign of achievement is stored digitally on a server.

1.2 The History of Badges

In 1911 boy scouts and girl scouts introduced merit badges. These merit badges were created in order to encourage scouts to explore areas, which interest them and teach them skills in scout craft. (Boy Scouts of America, 1912)

The badges were used as a proof of the scout's achievement in gaining knowledge in a specific area corresponding to the badge. Until the 1960's badges could be earned in any order as long as the scout met the specific requirements. Since the 1960's scouts are required to earn the rank of first class scout before they can start earning badges. (Boy Scouts of America, 2014)

The merit badge program has continued to grow to this day, recently there have been new badges created to recognize skills in game design, robotics, movie making, mining and programming. (Henning, 2014)

In the 1980's games began to implement 'levelling' as a form of game progression. Levelling is a similar concept to badges in that it allows a player to show off their achievements in a game. Unlike badges levelling is a logical progression, it is impossible for a player to skip over the easy levels and go straight to the highest level. Typically as gamers play through the game they gain levels by completing objectives. Higher levels unlock more features and make the player more powerful, which keep the player interested, which in turn leads to higher levels and so new features. It's an extremely addictive and motivating cycle.

In the 2000's the game industry introduced 'achievements'. Achievements are simply a digital version of boy scouts merit badges used in order to recognize in game accomplishments. They are also commonly known as badges, trophies, awards, stamps or medals. Achievements have no effect on the player's gameplay; they provide no advantages or disadvantages for unlocking or not unlocking them. The management of achievements takes place outside of the game and allows players

FIGURE 1: SCOUT MERIT BADGES

to show off specific goals or tasks they have accomplished in the game. (Eranti, 2011)

Most recently in 2009, the Mozilla Foundation took the concept of achievements from the game industry and re-coined the term badges. They went to work on creating an open platform for badges to be used in order to recognize achievements in learning.

1.3 The Mozilla Foundation

The Mozilla Foundation is a non-profit organization that promotes openness, innovation and participation on the Internet. Best known for their Firefox web browser, they also have a wide range of other projects. They actively engage in educational efforts such as Mozilla web maker, which teaches digital skills and web literacy in an attempt to move people from using the web to creating the web. (Mozilla, 2014) They are also the developers of the Open Badges platform, which is the focus of this report.

2.0 Open Badges

Mozilla Open Badges seek to unlock the full potential of learning online and offline by providing a lightweight platform for educators to create and award badges. (Mozilla, 2014) There are three key components to the open badges platform; earning badges, issuing badges and displaying badges.

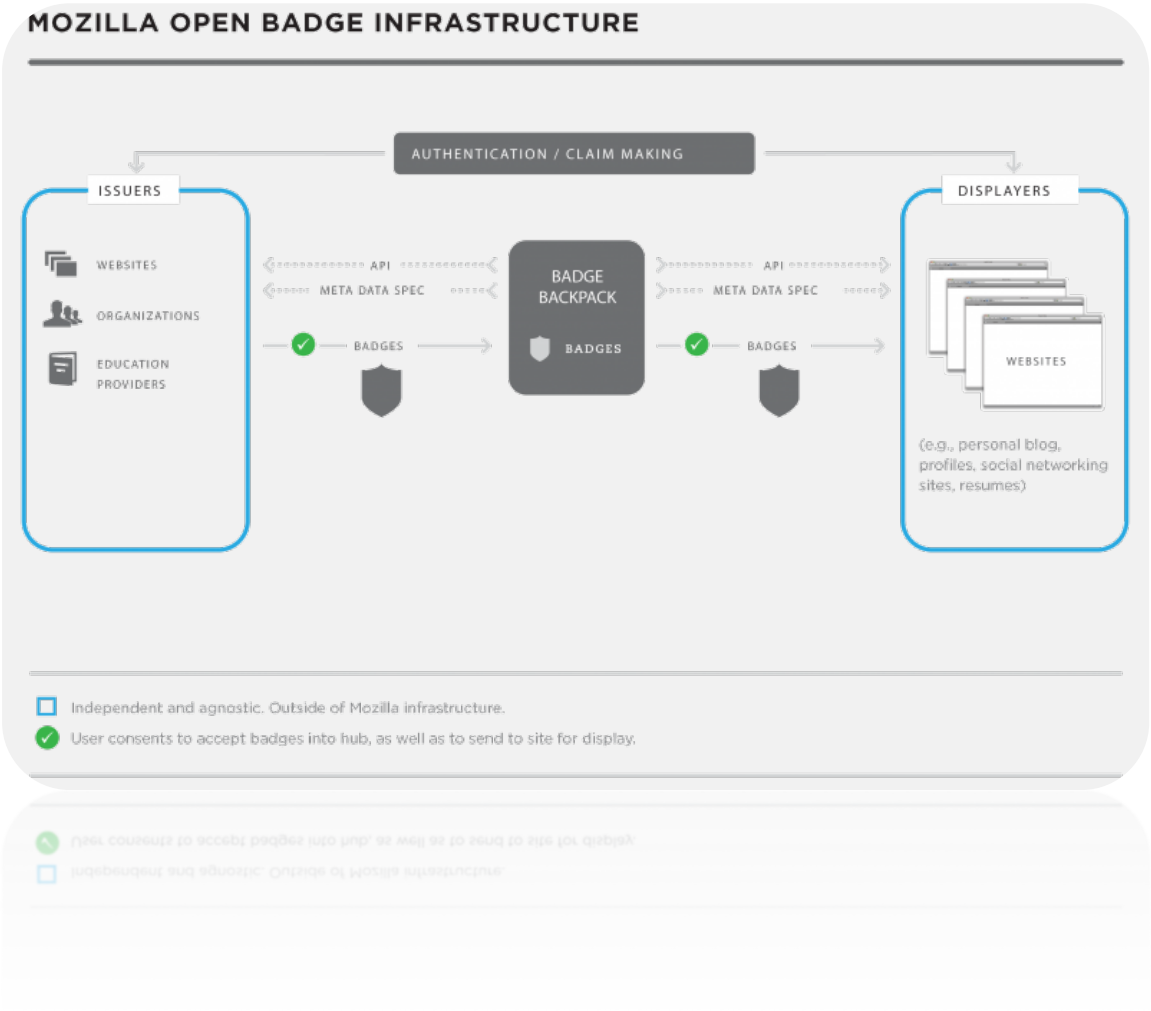

FIGURE 2: DIAGRAM ILLUSTRATING THE OPEN BADGES INFRASTRUCTURE

Figure 2 shows the typical communication channels between issuers, earners and displayers. The backpack acts are a middleman and provides APIs for both the displayer and the issuer, for this reason it is an extremely useful tool for earners to store their badges in the backpack.

2.1 BADGE CREATION

Creating an Open Badge requires creating a number of JSON assertions. An assertion may take two forms, either a hosted or a signed assertion. A hosted assertion is a set of JSON files hosted at a publicly accessible URL. These assertions must be served with the header type application/json, for example if using the apache web server adding to httpd.conf the following line will let it serve json files with the application/json header.

"AddType application/json json".

Use the open badges assertion validator[1] to validate badge assertions. The assertion is broken into three primary files, the badge class, the badge instance and the issuer organisation. A signed assertion is baked directly into the badge using a Java Web Signature (JWS) and is verified through the issuing websites public key.

[1] http://validator.openbadges.org

2.1.1 Assertion Objects & Properties[2]

The following tables present the JSON objects required for creating a Open Badge. The badge assertion, issuer organisation and badge class are the only compulsory objects with the other supporting objects available to be optionally referenced by the primary assertions.

TABLE 1: BADGE ASSERTION

Property	**Expected Type**	**Description**
uid	Text	Unique identifier for this badge instance. (must be unique on the issuing server)
recipient	**IdentityObject**	The recipient of the badge
badge	URL	URL linking to the hosted Badge Class which identifies the type of badge being awarded
verify	**VerificationObject**	Data to help a third party verify this assertion
issuedOn	DateTime	Date that the badge was issued
image	URL	URL of an image representing this users achievement. (Optional)
evidence	URL	URL of the work submitted that the recipient has completed to earn this badge. (Optional)
expires	DateTime	Date the badge expires and should no longer be considered valid. If left blank the badge will not expire. (Optional)

Table 1, badge assertion is required to create an open badge. The badge assertion acts as an individual instance of a badge and must have the properties of uid, recipient, badge, verify and issuedOn.

[2] https://github.com/mozilla/openbadges-specification/blob/master/Assertion/latest.md

TABLE 2: IDENTITY OBJECT

Property	**Expected Type**	**Description**
identity	IdentityHash or Text	Either the hash of the identity of the plaintext value.
type	IdentityType	The type of identity.
hashed	Boolean	Whether or not the identity value is hashed.
salt	Text	If the identity is hashed this should contain the string used to salt the hash. If this is left null it is assumed the has is not salted. (Optional)

The Identity Object, table 2, is referenced by the badge assertion for its recipient property. It is made up of an identity usually a users email address, either hashed or in plain text, the type of identity (usually "email") and boolean value indicating whether the identity has been hashed or not. The optional salt is included if used in the hash.

TABLE 3: VERIFICATION OBJECT

Property	**Expected Type**	**Description**
type	VerificationType	The type of verification method
url	URL	If the type is hosted this should be a URL pointing to the assertion on the issuer server. If the type is signed this should link to the issuers public key.

Table 3 shows the verification object. This is extremely simple and has only two propertyies, the type and url. The type is either "hosted" or "signed" if the type is hosted then the URL should point to the hosted badge assertion file, if it is signed it should point to the issuers public key. The verification object is the expected object in the badge assertion for the verify property.

TABLE 4: BADGE CLASS

Property	**Expected Type**	**Description**
name	Text	The name of the badge
description	Text	A short description of the badge
image	URL	URL of an image representing the badge
criteria	URL	URL of the criteria for earning the achievement
issuer	URL	URL pointing to the Issuer Organisation object representing the organisation who issued the badge.
alignment	Array of AlignmentObjects	List of objects describing which educational standards this badge aligns to if any. (Optional)
tags	Array of Text	List of tags that describe the type of achievement. (Optional)

The badge class shown in table 4 is a mandatory assertion in creating a badge. The badge class is typically pointed to by the badge assertions "badge" property as a URL. It consists of five required properties; name, description, image, criteria and issuer. The name and description are plain text properties, the image is a URL pointing to an image representing the badge, the criteria is a URL pointing to a page containing the criteria required to be met before being awarded the badge and the issuer is a URL pointing to the issuing organisations website. The optional properties of alignments and tags are arrays of alignment objects and text respectively. The alignment object properties are shown in table 5 below.

TABLE 5: ALIGNMENT OBJECT

Property	**Expected Type**	**Description**
name	Text	Name of the alignment.
url	URL	URL linking to the official description of the standard.
description	Text	Short description of the standard. (Optional)

TABLE 6: ISSUER ORGANISATION

Property	**Expected Type**	**Description**
name	Text	The name of the issuing organisation
url	URL	The URL of the organisations website.
description	Text	A short description of the organisation
image	URL	An image representing the organisation
email	Text	Contact email for someone at the organisation
revocationList	URL	URL pointing to the badge revocation list. Revocation List should be a JSON object where keys are the uid of revoked badge assertions and values are the reason for revocation. Only required for signed badges.

The final required component of the assertion is the Issuer Organisation, shown in table 6. The Issuer Organisation has two compulsory properties; name and url. The name is the name of the issuing organisation in plain text and the url is a url pointing to the issuing organisations website.

Optionally the Issuer Organisation object may also include a plain text description of the organisation, a link to an image representing the organisation, such as a logo, an email address to contact someone from the organisation and in the case that the organisation is issuing signed assertions a revocation list. The badge revocation list is a JSON list of badge uids which have been revoked and is shown in table 7 below.

TABLE 7: REVOCATION LIST

Property	**Expected Type**	**Description**
uid	Text	The uid of the badge being revoked
description	Text	A description of why the badge was revoked.

2.2 Mozilla Backpack

The Mozilla backpack acts as the middleman between badge issuing and badge displaying. The primary objective of the backpack is to provide a centralised location for storing badges received from any issuer. To do this, the Backpack records the URL of the badge assertion and uses this to verify the badges authenticity like any displayer. The backpack also provides a set of APIs that badge displayers may use to easily access a particular user's badges and display them and that issuing organisations may use to give earners the opportunity to save badges to their Backpack.

Users of the Backpack have total control over what badges stored in their Backpack may be accessed by displayers using the displayer API. They can organise their badges into groups and control the privacy settings for these groups. By setting a badge group to public anyone who knows the users email or Backpack ID number will be able to access and display the badges using the Backpack API.

Typically issuing organisations will provide the earner with the ability to push their badge to the Backpack by supplying a page implementing the Backpack issuer API. This will prompt the earner to log into their Mozilla persona backpack account and accept the badge. At this stage the Backpack also verifies the authenticity of the badge and checks that the earners email and the backpack account email addresses match up. This makes it impossible to add badges your own Backpack, which have been issued to another person.

2.2.1 BACKPACK ISSUER API[3]

The backpack provides a JavaScript API, which can be used to create a connection with a users backpack account and issue badges. The JavaScript file can be included with the following HTML script.

```
<script src="https://backpack.openbadges.org/issuer.js"></script>
```

This provides a function, which takes an array of assertion URLs and a callback function as parameters and from that creates a light boxed dialog prompting the users to login to their backpack and accept the badges being issued.

```
OpenBadges.issue(assertions, *callback*)
```

Example:

```
OpenBadges.issue([url1, url2], function(errors, successes) {
        //do something with the callback function
});
```

This will require the user to login and accept each badge, to create a persistent connection with the backpack owner the API provides a connect function which takes a URL to a callback page and a list of the scope of the connection (currently limited to 'issue', further control may become available in future).

```
OpenBadges.connect({
        callback: "https://issuer.org/callback",
        scope: ['issue']
});
```

[3] https://github.com/mozilla/openbadges/wiki/Backpack-Connect:-Issuer-Documentation

Once the user has authenticated the connection to their backpack by logging in, they will be redirected to the callback URL. The callback will be passed a number of GET parameters seen below.

TABLE 8: OPENBADGES.CONNECT CALLBACK GET PARAMETERS

GET Parameter	Description
error	if the user denied access to their backpack this will be set to 'access_denied'.
access_token	A string required to access the users backpack
refresh_token	A string used to acquire a new access_token when it expires.
expires	The number of seconds the access_token is valid before it needs to be refreshed
api_root	The absolute URL of the users backpack connect API endpoint.

Unless otherwise specified all API endpoints require authorization via the access_token. This is done using the Authorization HTTP header with the value set to 'Bearer b64_access_token' where b64_access_token is the Base64 encoding of the access_token. To issue a badge send POST request to *api_root/issue* with the following parameter.

TABLE 9: BACKPACK BADGE ISSUE POST PARAMETERS

POST Parameter	Description
badge	The absolute URL of the badge assertion to be issued.

When the access token has expired the issuer will receive the following response.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="backpack",
        error="invalid_token",
        error_description="The access token expired"
```

To refresh the token the issuer must send a POST request to *api_root/token*. This API endpoint doesn't require the authorization header since the access_token has expired The POST request must contain the following parameters.

TABLE 10: BACKPACK TOKEN REFRESH POST PARAMETERS

POST Parameter	**Description**
grant_type	Should be set to 'refresh_token'
refresh_token	The value of the refresh_token received in previous authorisation.

The response to the above POST request will contain a JSON string with the following keys.

TABLE 11: BACKPACK TOKEN REFRESH RESPONSE

JSON Response Key	**Description**
expires	The number of seconds the access_token is valid before it needs to be refreshed.
access_token	The new access_token string, required to access the users backpack.
refresh_token	The new refresh_token string, used to refresh the access_token when it expires again.

2.2.2 BACKPACK DISPLAYER API[4]

The backpack displayer provides a REST API with queries to convert a users email to their backpack userId, retrieve a list of a users public badge groups and retrieve the badges in a public group.

To convert a users email to their backpack userId the displayer should send a POST request to *http://backpack.openbadges.org/displayer/convert/email* with the following parameter.

TABLE 12: BACKPACK EMAIL CONVERT POST PARAMETER

POST Parameter	**Description**
email	The users email to convert to backpack user id

The server will respond with a JSON object, which if successful will contain the userId.

To retrieve the list of public badges for a particular userId a GET request must be sent to *http://backpack.openbadges.org/displayer/:userId/groups.json* where :userId is the user whose badge groups are being retrieved. The response is a JSON object with the userId and a list of badge groups each identified by a 'groupid'. (example response seen below).

```
{
'userid' : 1,
'groups': [{ 'groupid': 1,
        'name'    : "Groups Name",
        'badges' : 10
        }]
}
```

[4] https://github.com/mozilla/openbadges/wiki/Displayer-API

To retrieve the badges in a public badge group for a particular userId a GET request must be sent to *http://backpack.openbadges.org/displayer/:userId/group/:groupid.json* where userId is the user whose badges are being retrieved and groupid is the group of badges being retrieved for that user. (example response seen below)

```
{
'userid'   : 1,
'groupid' : 1,
'badges'   : [{ 'badge assertion goes here' }]
}
```

2.3 Issuing, Earning & Displaying Badges

2.3.1 Issuing

To issue a badge, first the issuing organization must create the badge class, issuer organisation and badge assertion JSON files and have them hosted on their server. The issuer may choose to stop there, by providing the badge assertion URL to the earner they will have awarded the badge. Typically they will provide means of assisting the user in pushing the badge to their backpack. In this case the issuer uses the backpack issuing API to create a web page for the user to log in and add the badges to their backpack.

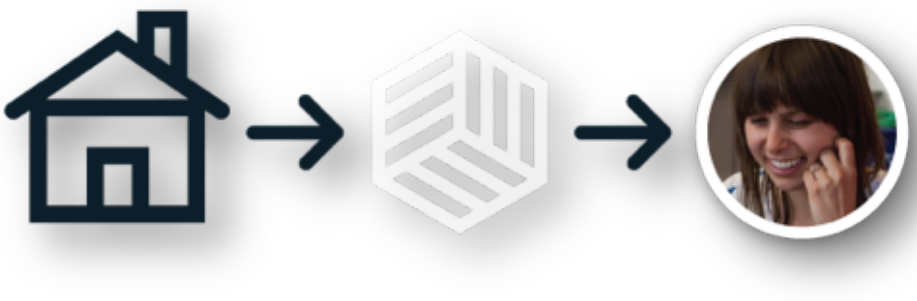

FIGURE 3: ISSUING BADGES

2.3.2 Earning Badges

Earning a badge is a simple process, first a student must display to the badge issuer (the course instructor in most cases) that they satisfy the requirements for being awarded a particular badge. The issuer can then issue a badge to the student through their organisations badge issuing system. The student is required to then accept the badge by going to their Mozilla backpack and accepting the awarded badge. This is a safety measure to avoid backpacks being spammed with badges.

FIGURE 4: EARNING BADGES

2.3.3 Displaying Badges

Using the Mozilla backpack badges can be sorted into collections and shared across professional and social networks, such as LinkedIn, twitter or a blog. Alternatively they can also be embedded directly into a resume or email.

In order to share and display badges, a user must first log in to their backpack and create a "badge group", set this group to be publicly viewable and then click the sharing icon on the bottom right of the group.

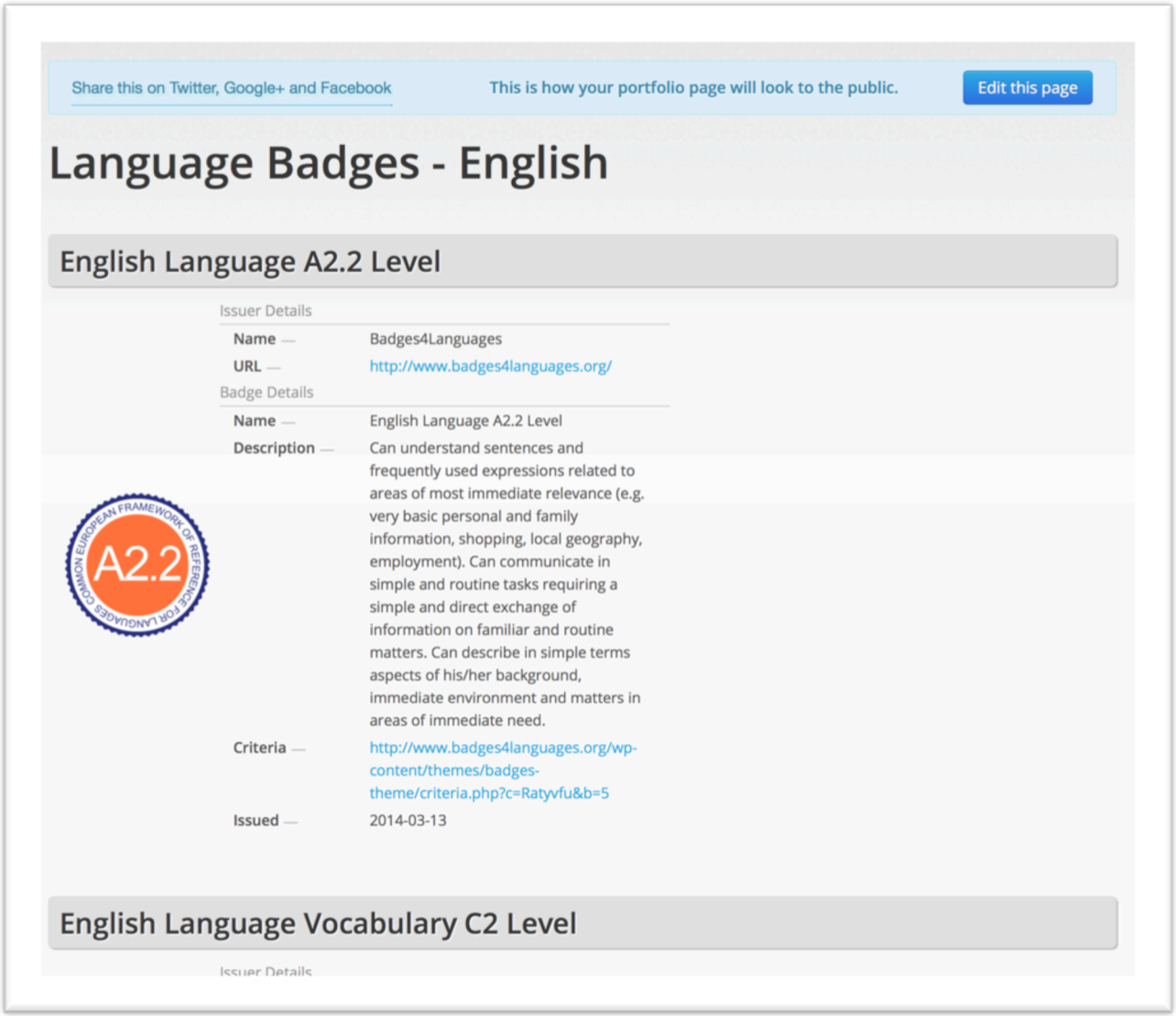

FIGURE 5: BACKPACK PUBLIC PORTFOLIO

As seen in figures 5 and 6, the backpack allows users some customization of the way the badge is displayed. They are also able to add extra information about the badge and a subtitle in order to further identify similar badges. Unless the user links directly to this display it is still up to the displayer site as to whether or not this additional information is displayed.

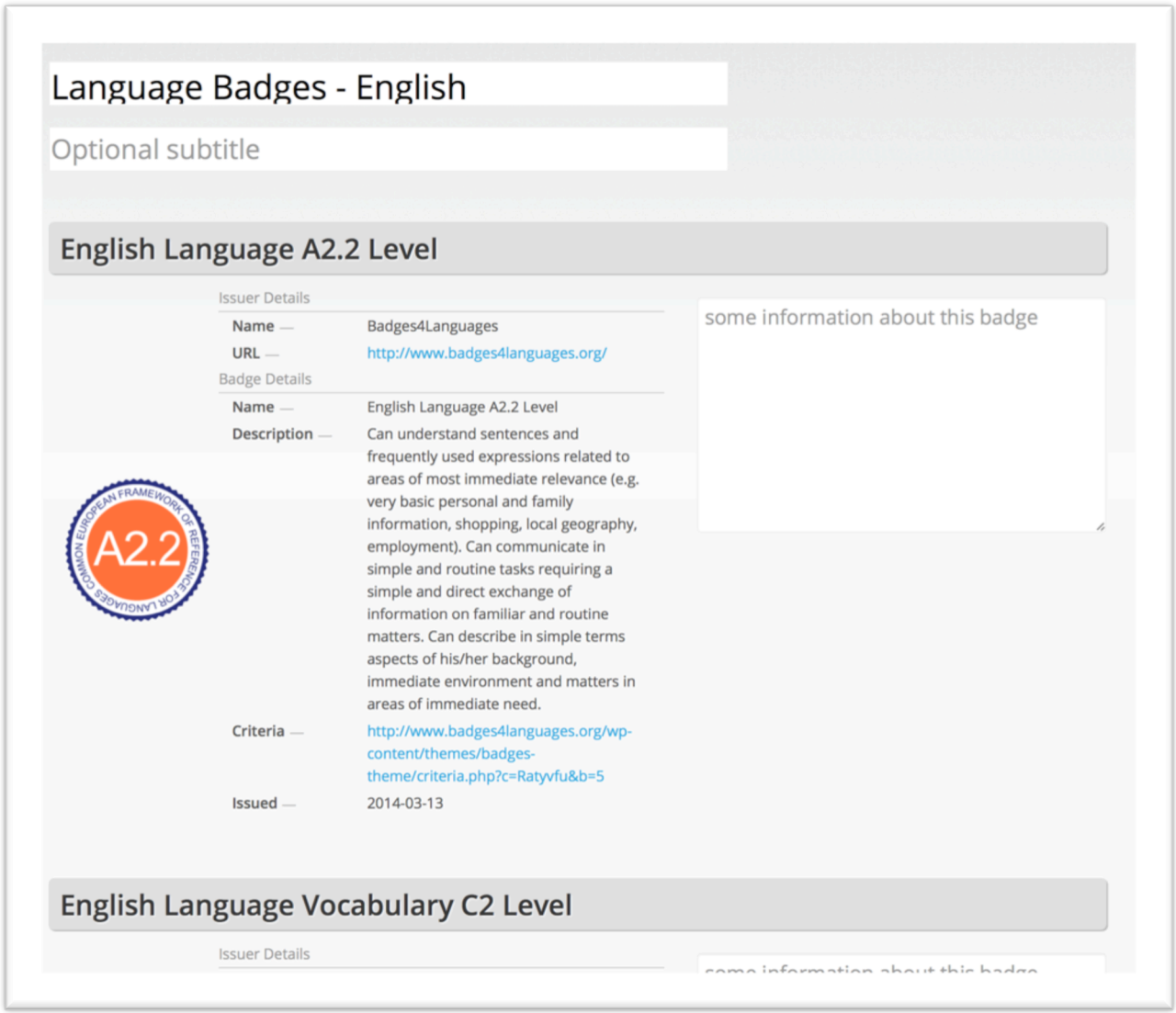

FIGURE 6: BACKPACK PUBLIC PORTFOLIO, EDITING OPTIONS

3.0 OPEN BADGES BADGEKIT

While investigating Mozilla's Open Badges, it became necessary to have a simplistic way of quickly creating and issuing new badges. This problem led to the creation of the Open Badges Badge Kit (BadgeKit). Built with NodeJS, ExpressJS and MySQL the BadgeKit allows a user to quickly create view and issue new badges using a web app.

The BadgeKit is broken down into two key applications. The BadgeKit-API directly manages the storage and organization of badges, badge instances, issuer information and all of the other information required by the Open Badges standard. The BadgeKit also offers a front-end application, which creates the web app allowing for a user to create, issue and manage the badges. This app communicates with the BadgeKit-API in order to make changes to the badge database.

3.1 BADGEKIT TECHNOLOGIES & INSTALLATION

The BadgeKit and the BadgeKit-API is built using Node.JS & ExpressJS with a MySQL database. See the associated installation guide for instructions on how to setup and install the BadgeKit and BadgeKit-API.

3.1.1 NODEJS

Node.JS is an event driven programming language, built on top of Google Chrome's V8 JavaScript runtime engine. It allows for server side applications to be programmed and run using JavaScript. In terms of speed and performance it is extremely fast because it runs on an event loop, this means all it's functions are non-blocking. It is also very light weight and efficient, this makes it a great choice for writing web APIs and web applications. The only downside is that due to the event driven nature it is a bit more challenging to understand program flow than a more basic language.

3.1.2 EXPRESSJS

Express is to Node as Rails is to Ruby. It provides a minimal and flexible web application framework. This allows Node web apps to be quickly developed and deployed to the web. Express provides a range of HTTP methods and middleware as well as allowing a Model-View-Controller (MVC) setup to be easily created.

3.1.3 MySQL

MySQL is used as the database storage system of choice for both the BadgeKit-API as well as the front end BadgeKit application. MySQL is the worlds most popular SQL database management system. The following is an overview of the database schema for the BadgeKit and the BadgeKit-API.

The BadgeKit database stores assertion information for badge classes created. The BadgeKit API stores the badge instances for every awarded badge as well as the classes and issuer organisations for each badge.

3.2 BadgeKit API Usage

The BadgeKit uses a REST API and allows users to retrieve, update or add; new badges, earner applications, application reviews, issuers within a system, programs within an issuer, and badge/application data for particular issuers or programs. All API requests return JSON data, which can then be used and represented on the issuer site as desired. Each request also requires a shared secret to be used for authentication purposes. Further to making http POST / GET request the API provides the ability to setup a web hook URL to handle badge application and badge issuing notifications.

3.2.1 Claims & Authorization

Claims are authorized using a JWT for each request and putting it in the HTTP header. The token should use the HS256 (HMAC-SHA256) algorithm.

The claim payload data requires a number of parameters to be filled, these are shown in table 13.

TABLE 13: BADGEKIT API CLAIM PARAMETERS

Payload Parameter	**Description**
key	At this time only "master" is supported, in future the API will support more keys for different consumers.
exp	Unix Time when this token should expire. (Optional)
method	HTTP Method being used (GET, POST, PUT, DELETE)
path	HTTP path / endpoint being requested, should include any query variables.
body	This is only required for POST / PUT requests. It consists of a JSON object containing two parameters, **alg** and **hash**. At this time only sha256 is supported for the alg and the hash should be the hash of the entire POST or PUT request being made.

Example claim in JavaScript, requests all badge instances from the badgekit system with the email craig.knott@uqconnect.edu.au.

```
header: {typ: 'JWT', alg: 'HS256'},
  payload: {
    key: 'master',
    exp: Date.now() + (1000 * 60),
    method: 'GET',
    path: '/systems/badgekit/instances/craig.knott@uqconnect.edu.au'
},
secret: 'yoursecret'
```

3.2.2 API ENDPOINTS

The API has implemented a huge number of endpoints, catering to any request that could be made. Each endpoint starts with the /systems/[slug] path, except for two which will GET or POST to all the systems in the API database using the endpoint /systems. Otherwise the system slug must be included in the path. From this point, either continue adding levels of management systems for instance /systems/[slug]/issuers/[slug]/programs/[slug]. Once the required management level has been reached (for example if your system had issuers, but no programs within the issuers) then the endpoints start branching off to the data to be interacted with, eg: badges, claim codes, badge instances, badge applications, or milestones.

For a full list of the BadgeKit API endpoints, see Appendix A

5.0 DISPLAYING BADGES IN SPACES

The ability to view large clusters of badges and gain an idea of someone's strengths and weaknesses based on their badge clusters is a problem which has arisen with the creation of badges.

Assuming a user groups their badges appropriately, grouping badges by area the badge relates to, the badges can be displayed in clusters allowing a viewer to easily relate large clusters to greater knowledge in a particular area.

Another visualization badges may be used for is to view the strengths and weaknesses of a team. To do this a model of the team's work place should be created and a link made to each of the team members backpacks within the model. The badges would then be imported from the users backpack and displayed within the 3D space. A viewer may then easily observe the differences in skills represented by badges among team members by walking through the space.

The next question that arises is how would we go about creating these interesting visualizations and displays involving badges? For this research the Unity3D game engine has been used to create a simple set of assets, which can easily be used to create new badge displays. The following sections provide an overview of the Unity3D engine, how the open badges asset package was created, what it does and how it can be used and extended.

5.1 About Unity 3D

Unity3D is a powerful cross-platform gaming and game development engine. It is currently the most popular game development platform on the market due to its abundance of features, which greatly simplify the process of building games. Unity can be downloaded from the unity website[5] (figure 7) and is available in a free and pro version. The pro version offers all the functionality and features, while the free version only provides the basic features. All work done using the unity engine has been completed using the pro version.

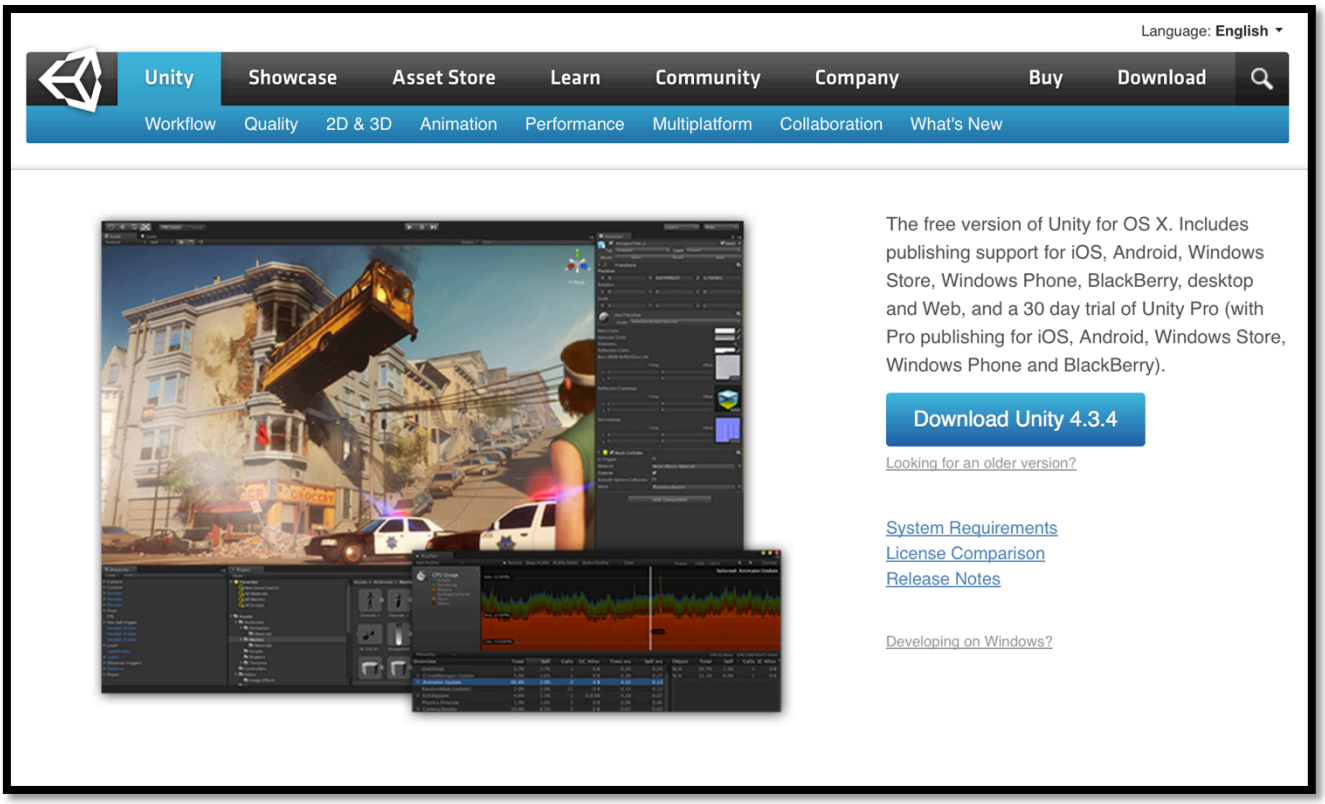

FIGURE 7: UNITY3D WEBSITE

Unity has all the bells and whistles that have come to be expected of any top end game development engine. It uses DirectX / Direct3D for rendering on windows, OpenGL for Mac and Linux, and OpenGL ES for iOS and Android. It provides support for a wide range of different texture mapping techniques such as reflective, parallax and bump mapping as well as dynamic shadows and advanced lighting techniques. Unity handles imports of 3D objects from any modern 3D modelling or CAD applications such as 3DS Max and Maya.

[5] https://unity3d.com/download

Scripting in unity is built on Mono (the open source .NET implementation) and is written in either C#, Unity Script (a language similar to JavaScript in syntax but with slight differences), or Boo (a language similar to python).

Unity has built in support for NVidia's PhysX engine providing highly realistic physics, it also has added support for real time cloth simulation and collision layers. Recently support for the 2D physics engine, Box2D, has also been added.

Probably the greatest feature of the Unity3D game engine is its ability to develop a game once and then publish it to any number of different platforms with the click of a button. Unity supports publishing to Windows, Mac, Linus, iOS, Android, Xbox, PlayStation, Blackberry, Windows Phone, Wii and Unity Web Player.

5.2 Open Badges Asset Package

The Open Badges assets designed for the Unity game engine were created to be highly modular, but still extremely easy to implement. The package consists of 3 linked prefabs. The user prefab (seen in figure 8) is the root user and requires a users email address linked to a Mozilla Persona and backpack in order to instantiate the badge groups and badge prefabs.

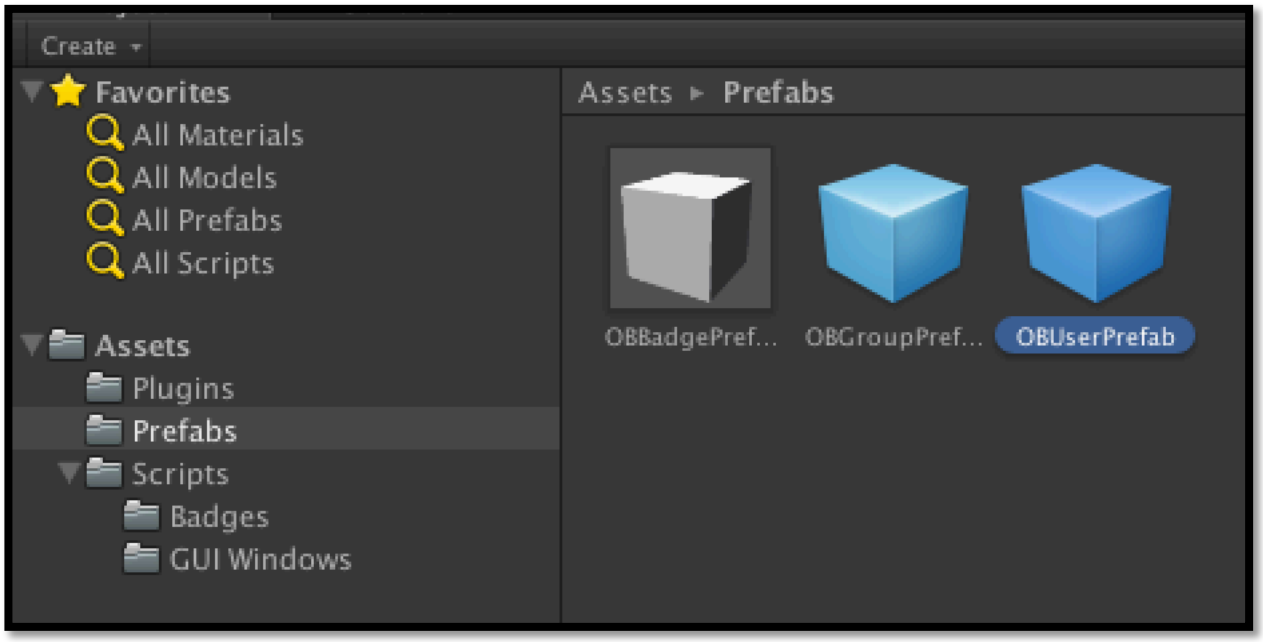

FIGURE 8: ASSET PACKAGE PREFABS

The user prefab will create new badge groups and for each badge group all the badges within that group will be generated, have the appropriate badge textures downloaded and applied, and the badge information stored within the badge object. The badge objects are then visualised using textured cubes as seen in figure 9 and further detail can be gained by clicking on an individual badge, seen in figure 10.

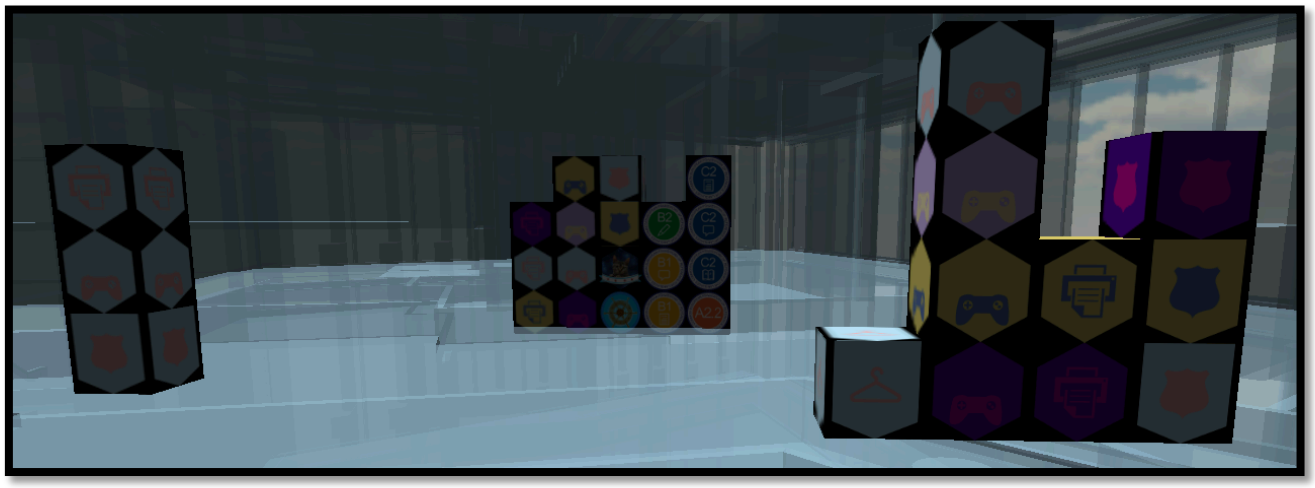

FIGURE 9: ASSET PACKAGE BADGE VISUALISATION

For installation instructions see the attached installation guide, however it is as simple as importing the package into unity and then using it. One example demonstration using the asset package was created and in order to prove its functionality.

FIGURE 10: ASSET PACKAGE BADGE DETAILS

5.3 Asset Package Demonstration

To demonstrate the asset package it has been used to visualise a number of users and their badges in their work places around the learning innovation building. In order to do this first a CAD model of the building supplied by the building architect was imported into the game engine and scaled to an appropriate size (figure 11).

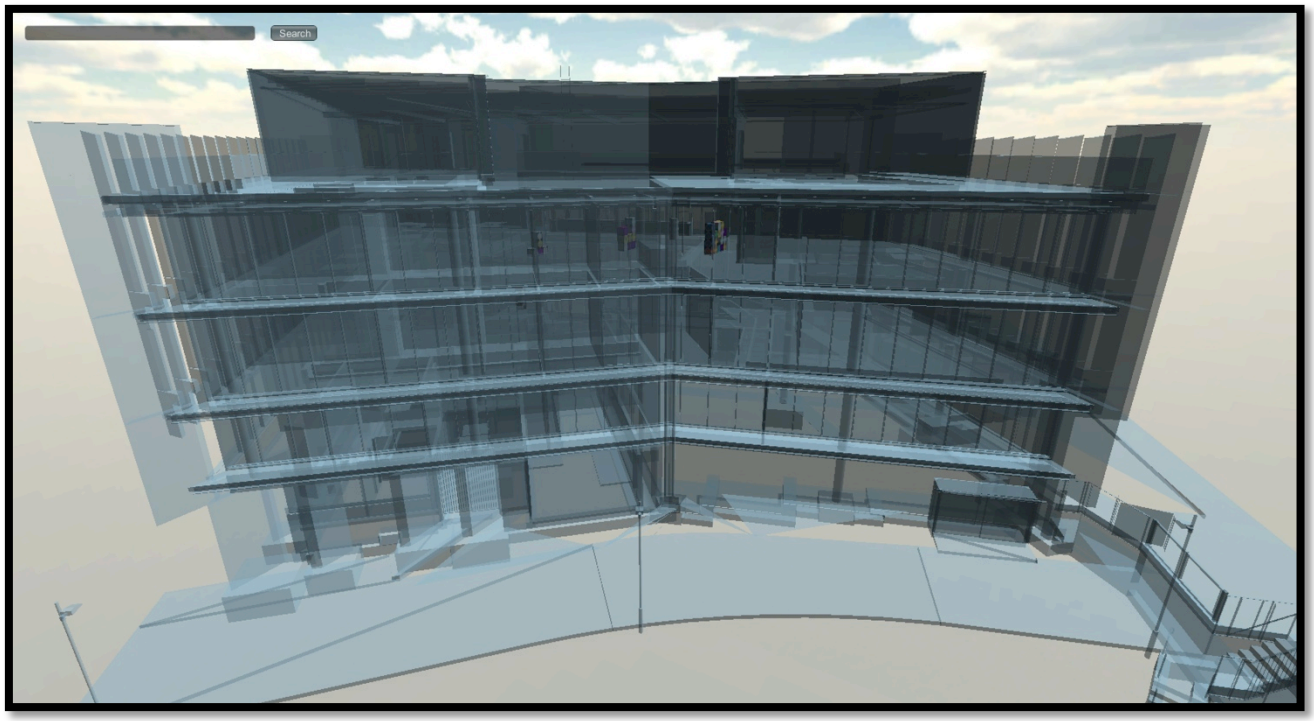

FIGURE 11: OPEN BADGES DEMO LEARNING INNOVATION BUILDING

Then the asset package was imported and used to place some users and their badges in the building as appropriate. Some additional scripts were added in order to further improve the asset package by providing simple search functionality (figure 12).

FIGURE 12: OPEN BADGES DEMO SEARCH FEATURE

Once a user has typed in some key words for searching in the search bar and clicked search they will be presented with a list of results from all the badges imported into the demonstration (figure 13). Clicking on one of the results will move the view to be centred on that specific badge.

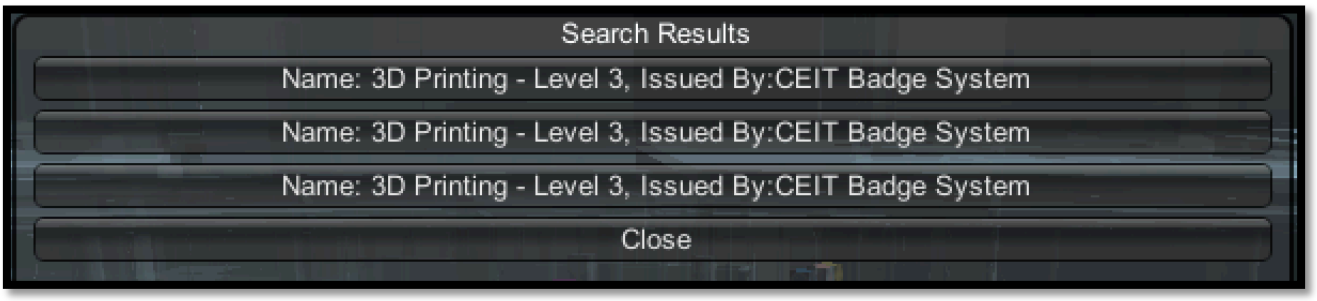

FIGURE 13: OPEN BADGES DEMO SEARCH RESULTS

The end result was a demonstration allowing a user to come to the demo search for some key words and be presented with links to anyone in the demonstration who has badges meeting the search criteria. Alternatively the users are able to explore the visualisation and click on individual badges to learn more about them.

6.0 REFERENCES

Badges For Languages. (2014). *Badges for Languages*. Retrieved April 12, 2014 from Badges for Languages: http://www.badges4languages.org

Boy Scouts of America. (1912). *Handbooks for Boys.* Michigan: Boy Scouts of America.

Boy Scouts of America. (2014). *Introduction to Merit Badges*. Retrieved March 23, 2014 from Boy Scouts of America: http://www.scouting.org/meritbadges.aspx

City of Chicago. (2013). *CSOL | What are awesome things to do in Chicago this Summer?* Retrieved April 28, 2014 from Chicago Summer of Learning: http://www.chicagosummeroflearning.org

Dictionary.com. (2014). *Badge*. Retrieved April 1, 2014 from Dictionary.com: http://dictionary.reference.com/browse/badge?s=t

Duersch, F. (2003). *Merit Badge Field Guide.* Utah: Downs Printing, Inc.

Eranti, V. (2011). *Framework for Designing and Evaluating Game Achievements.* Retrieved March 22, 2014 from Digital Games Research Association: http://www.digra.org/wp-content/uploads/digital-library/11307.59151.pdf

Henning, S. (2014). *Merit Badges, Past and Present, and Their Evolution*. Retrieved March 23, 2014 from Henning's Scouters' Pages: http://www.scouters.us/mb.php#anchorTOP

IMS Global Learning Consortium, Inc. (2014). *IMS Global: Learning Tools Interoperability*. Retrieved March 22, 2014 from IMS Global: http://www.imsglobal.org/toolsinteroperability2.cfm

Manning, C. (2013, August 5). *Open Badges for Schools - What are the options?* Retrieved April 22, 2014 from Makewaves Blog: http://blog.makewav.es/2013/08/05/open-badges-for-schools-what-are-the-options/

Mozilla. (2014). *Backpack Displayer API*. Retrieved March 12, 2014 from GitHub: https://github.com/mozilla/openbadges/wiki/Displayer-API

Mozilla. (2014). *Backpack Issuer API*. Retrieved March 12, 2014 from GitHub: https://github.com/mozilla/openbadges/wiki/Backpack-Connect:-Issuer-Documentation

Mozilla Foundation. (2012, December 18). *RFC: Badge Validation.* Retrieved May 25, 2014 from RFC: An Open, Distributed System for Badge Validation: https://docs.google.com/file/d/0BwJ_PQhV0lJTSnYtQzV5Q0FxNDA/edit

Mozilla. (2014). *Home of Mozilla*. Retrieved March 3, 2014 from Mozilla: https://www.mozilla.org

Mozilla. (2014). *Mozilla Open Badges*. Retrieved March 3, 2014 from Open Badges: http://www.openbadges.org

Mozilla. (2014). *Open Badges Assertions*. Retrieved March 12, 2014 from GitHub: https://github.com/mozilla/openbadges-specification/blob/master/Assertion/latest.md

Netting, R. (2011, April 12). *NASA Science*. Retrieved March 21, 2014 from NASA and BSA Introduce Robotics Merit Badge: http://science.nasa.gov/science-news/news-and-features/nasa-and-bsa-introduce-robotics-merit-badge/

Peer 2 Peer University, The Mozilla Foundation. (2012, August 27). *Open Badges for Lifelong Learning.* Retrieved April 12, 2014 from Mozilla: https://wiki.mozilla.org/images/5/59/OpenBadges-Working-Paper_012312.pdf

Reconnect Learning. (2014). *Badges for Lifelong Learning.* Retrieved May 25, 2014 from Case Studies | Reconnect Learning: http://cloudfront.sproutfund.org/downloads/reconnectlearning/DML4.pdf

Reconnect Learning. (2014). *Chicago Summer of Learning.* Retrieved May 25, 2014 from Case Studies | Reconnect Learning: http://cloudfront.sproutfund.org/downloads/reconnectlearning/CSOL.pdf

The Mozilla Foundation. (2014). *Mozilla Backpack*. Retrieved March 7, 2014 from Mozilla Backpack: https://backpack.openbadges.org/backpack

Unity Technologies. (2014). *Unity - Game Engine*. Retrieved November 20, 2013 from Unity - Game Engine: http://unity3d.com

Unity Technologies. (2014). *Unity - Manual: Shaders*. Retrieved June 1, 2014 from Unity3D - Game Engine: http://docs.unity3d.com/Manual/Shaders.html

Wikipedia. (2014). *History of Merit Badges*. Retrieved March 20, 2014 from Wikipedia: http://en.wikipedia.org/wiki/History_of_merit_badges_(Boy_Scouts_of_America)

7.0 APPENDIX A – BADGEKIT API ENDPOINTS[6]

CONTAINERS

SYSTEMS

GET /systems

POST /systems

GET /systems/:slug

PUT /systems/:slug

DELETE /systems/:slug

ISSUERS

GET /systems/:slug/issuers

POST /systems/:slug/issuers

GET /systems/:slug/issuers/:slug

PUT /systems/:slug/issuers/:slug

DELETE /systems/:slug/issuers/:slug

PROGRAMS

GET /systems/:slug/issuers/:slug/programs

POST /systems/:slug/issuers/:slug/programs

GET /systems/:slug/issuers/:slug/programs/:slug

PUT /systems/:slug/issuers/:slug/programs/:slug

DELETE /systems/:slug/issuers/:slug/programs/:slug

[6] https://github.com/mozilla/badgekit-api/blob/master/docs/api-endpoints.md

Badge Management

Badges

GET /systems/:slug/badges

GET /systems/:slug/issuers/:slug/badges

GET /systems/:slug/issuers/:slug/programs/:slug/badges

POST /systems/:slug/badges

POST /systems/:slug/issuers/:slug/badges

POST /systems/:slug/issuers/:slug/programs/:slug/badges

GET /systems/:slug/badges/:slug

GET /systems/:slug/issuers/:slug/badges/:slug

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug

PUT /systems/:slug/badges/:slug

PUT /systems/:slug/issuers/:slug/badges/:slug

PUT /systems/:slug/issuers/:slug/programs/:slug/badges/:slug

DELETE /systems/:slug/badges/:slug

DELETE /systems/:slug/issuers/:slug/badges/:slug

DELETE /systems/:slug/issuers/:slug/programs/:slug/badges/:slug

Claim Codes

GET /systems/:slug/codes/:code

GET /systems/:slug/issuers/:slug/codes/:code

GET /systems/:slug/issuers/:slug/programs/:slug/codes/:code

GET /systems/:slug/badges/:slug/codes

GET /systems/:slug/issuers/:slug/badges/:slug/codes

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes

POST /systems/:slug/badges/:slug/codes

POST /systems/:slug/issuers/:slug/badges/:slug/codes

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes

POST /systems/:slug/badges/:slug/codes/random

POST /systems/:slug/issuers/:slug/badges/:slug/codes/random

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes/random

GET /systems/:slug/badges/:slug/codes/:code

GET /systems/:slug/issuers/:slug/badges/:slug/codes/:code

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes/:code

DELETE /systems/:slug/badges/:slug/codes/:code

DELETE /systems/:slug/issuers/:slug/badges/:slug/codes/:code

DELETE /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes/:code

POST /systems/:slug/badges/:slug/codes/:code/claim

POST /systems/:slug/issuers/:slug/badges/:slug/codes/:code/claim

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/codes/:code/claim

ISSUING

GET /systems/:slug/instances/:email

GET /systems/:slug/issuers/:slug/instances/:email

GET /systems/:slug/issuers/:slug/programs/:slug/instances/:email

GET /systems/:slug/badges/:slug/instances

GET /systems/:slug/issuers/:slug/badges/:slug/instances

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/instances

GET /systems/:slug/badges/:slug/instances/:email

GET /systems/:slug/issuers/:slug/badges/:slug/instances/:email

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/instances/:email

POST /systems/:slug/badges/:slug/instances

POST /systems/:slug/issuers/:slug/badges/:slug/instances

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/instances

DELETE /systems/:slug/badges/:slug/instances/:email

DELETE /systems/:slug/issuers/:slug/badges/:slug/instances/:email

DELETE /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/instances/:email

ASSESSMENT

GET /systems/:slug/applications

GET /systems/:slug/issuers/:slug/applications

GET /systems/:slug/issuers/:slug/programs/:slug/applications

GET /systems/:slug/badges/:slug/applications

GET /systems/:slug/issuers/:slug/badges/:slug/applications

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications

POST /systems/:slug/badges/:slug/applications

POST /systems/:slug/issuers/:slug/badges/:slug/applications

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications

PUT /systems/:slug/badges/:slug/applications/:slug

PUT /systems/:slug/issuers/:slug/badges/:slug/applications/:slug

PUT /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug

GET /systems/:slug/badges/:slug/applications/:slug

GET /systems/:slug/issuers/:slug/badges/:slug/applications/:slug

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug

DELETE /systems/:slug/badges/:slug/applications/:slug

DELETE /systems/:slug/issuers/:slug/badges/:slug/applications/:slug

DELETE /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug

GET /systems/:slug/badges/:slug/applications/:slug/reviews

GET /systems/:slug/issuers/:slug/badges/:slug/applications/:slug/reviews

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug/reviews

GET /systems/:slug/badges/:slug/applications/:slug/reviews/:slug

GET /systems/:slug/issuers/:slug/badges/:slug/applications/:slug/reviews/:slug

GET /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug/reviews/:slug

POST /systems/:slug/badges/:slug/applications/:slug/reviews

POST /systems/:slug/issuers/:slug/badges/:slug/applications/:slug/reviews

POST /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug/reviews

PUT /systems/:slug/badges/:slug/applications/:slug/reviews/:slug

PUT /systems/:slug/issuers/:slug/badges/:slug/applications/:slug/reviews/:slug

PUT /systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug/reviews/:slug

DELETE /systems/:slug/badges/:slug/applications/:slug/reviews/:slug

DELETE /systems/:slug/issuers/:slug/badges/:slug/applications/:slug/reviews/:slug

DELETE
/systems/:slug/issuers/:slug/programs/:slug/badges/:slug/applications/:slug/reviews/:slug

MILESTONES

GET /systems/:slug/milestones

POST /systems/:slug/milestones

GET /systems/:slug/milestones/:milestoneId

PUT /systems/:slug/milestones/:milestoneI

DELETE /systems/:slug/milestones/:milestoneId