Team DigiLearn

Technological Feasibility:

The Digital Backpack

Doctor Morgan Vigil-Hayes
Volodymyr Saruta
Caitlin Abuel
Grace Shirey
Israel Bermudes
Kristine Hermosado
Sebastian Kastrul

10 October, 2020

Team DigiLearn

Table of Contents

Team DigiLearn

1.0 - Introduction

The COVID-19 pandemic has led to a sudden shift to remote learning. Unfortunately, many students across America don't have access to a reliable Internet connection. The phenomenon known as “the homework gap” affects nearly 12 million students that are unable to fully participate in their coursework due to a lack of sufficient Internet access. Such a situation disproportionately affects disenfranchised communities. These students must rely on public hotspots to complete their assignments. In addition, students may spend a significant amount of their limited time trying to plan around the available connectivity, rather than on learning.

Team DigiLearn is working with Dr. Vigil Hayes and CANIS labs to bring to life The Digital Backpack. The Digital Backpack, or DigiPack is an app that will allow a fluid transition between online and offline learning. When a user comes into range of a Wi-Fi connection, the DigiPack will automatically download requested content for offline use later. The app will also automatically upload completed assignments for the user. These upload and download requests can be queued offline to be performed when a network connection is available. The app will interface with popular Learning Management Systems such as Google Classroom.

This document serves to examine technological challenges that may arise during the development of this software. Section 2 begins by introducing the main technological challenges for the project. Section 3 expands on these challenges with a detailed analysis of the technologies related to each specialized area. The ideas are aggregated in Section 4, wherein each individual technology comes together to form the planned system. The final section, Section 5, concludes the feasibility document with a review of the project goals and a summary of the system.

Team DigiLearn

2.0 - Technological Challenges

This section outlines the main technical challenges that may be faced for this project. These challenges relate to the major design aspects necessary to create a working product for the client. Each subsection discusses a technology, its purpose for the project, and the requirements related to that technology.

2.1 User Interface

2.1.1 Purpose

The DigiPack's goal is to provide offline service to K-12 students who have limited internet access. The application serves as a place where students can pull and work on online assignments without an internet connection.

2.1.2 Requirements

Considering that the users of the application would be K-12 students, it is essential to have different layouts for each age group. One of the tasks for the project is to have various user interfaces that will accommodate users' preferences. The following design challenges are brought up as the challenges that might occur while completing the project:

Easy to Use - With the different age groups, elementary students' cognitive ability is not as prominent as high school students. The system will need to make a UI (user interface) design that must be able to satisfy elementary students and high school students' expectations. With this in mind, the tool needs to enable an efficient design flow by making feature navigation simple.

Collaboration Friendly - Since the application will be based on K-12 students (users), there is a need for multiple visually appealing and informative designs for elementary/middle school and high school students. So, having an application that allows for collaboration on the same design file would be beneficial.

Compatibility - The application will be implemented in Android and iOS operating systems, which would mean designs based on Android and iOS would be required. A tool that allows cross-platform designs and is able to make prototypes would be helpful to the progress of the project.

2.2 Cross Platform Development

2.2.1 Purpose

The Digital Backpack is a mobile and web application used to service students while online and offline, mainly students from K-12. Since this project targets a wide audience, the cross platform development tool should be able to be accessed by multiple platforms and still communicate well with the user and run responsively.. Downloading assignments and queries can be a difficult task while offline, which is why the code used for the Digital Backpack needs to be efficient with resources and be sensibly designed so that the code is easy to update and debug. The application needs to behave reliably at all times. The solution is further discussed in 2.3.3.

2.2.2 Requirements

The Digital Backpack is an application aimed to help students across the nation. The application needs to be able to have the following features so that the application looks and runs perfectly for different platforms:

Simple Architecture - Since the Digital Backpack is going to be used by students and teachers, it is important that the UI architecture is intuitive and effective. The application should be easy to use for both sides. It is desirable to have a tool that allows for the creation of an application that the users can easily read and follow on for multiple platforms.

Reaches Across Multiple Platforms - There is a wide variety of people that will be using this application. There are many platforms that are being used these days. The goal is to be able to help students and in order to do that the Digital Backpack needs a tool that is able to reach across multiple platforms. The main platforms under consideration are IOS, Android, and Web Applications. The mobile application requires a Framework that is flexible with all of these platforms.

Functions Fast and Without Errors- The Digital Backpack is going to be used by students very often for help. In order to assist students efficiently , the application needs to be efficient and responsive, especially since the main function of the application is to be used while offline. The code cross platform development tool must run fast in order to provide the most helpful service to the user.

Easy to Fix Bugs and Update - Technology is moving at a rapid pace and the Digit Backpack needs to stay up-to-date. The code needs to be structured to allow for the efficient identification of bugs and straight-forward updating of the application.

2.3 Networking

2.3.1 Purpose

The Digipack must be able to network with educational services and retrieve data for the user. Content will be downloaded opportunistically, so it is essential that this transaction can occur as quickly and seamlessly as possible. A proxy server can be used for the application to aggregate data related to school content while the user is offline. This pre-fetched data will ensure fast download speeds when a user reaches an opportunistic connection. The solution is further discussed in section 3.3.

2.3.2 Requirements

There will be a proxy server that will aggregate data related to school content while the user is offline. A constraint from the client is that the proxy server should be Linux-based, and the service should be hosted using cloud services. The working prototype will be a cloud-based server that can serve 25-50 students. The feature requirements of the proxy server are as follows:

Integrates with Google Classroom APIs - Google Classroom (GC) is a popular LMS utilized by many teachers across america. The application should be able to interface with GC in order to perform important academic tasks such as: fetch announcements, assignments, and grades.

Integrates with Google Search - As a search engine, Google offers a vast array of knowledge from various sources. The application should be able to: store search queries to be posted when a connection is available, store top query responses for offline viewing.

Manages OAuth credentials - The Digital Backpack will be handling delicate data such as user information. As such it is essential to create a system that is secure and reliable. Further information about security is discussed in sections 2.5 and 3.5.

Provides a REST API - Data that is requested by the user may be coming from various different sources. The application must be able to collect this data and convert it to a uniform format that can be used within the Digital Backpack interface. Further information about REST services is discussed in sections 2.4 and 3.4.

Pre-fetches content requested by a user - One of the roles of the proxy server is to decrease the time necessary for download. Having common information cached allows content to quickly be pushed/pulled when opportunistic connectivity is established.

2.4 Integrating Multiple Resources

2.4.1 Purpose

The end user(s) of the Digital Backpack need to be able to access resources such as the ones described in section 3.4 in a quick and easy fashion while still conforming to the limitations of a mobile device. Student users will need to be able to access multiple web based educational resources and outlets all from the mobile application. Teacher users would like to be able to provide lists of resources that students can access, assignments, and possibly even curate the sources available to their students on a class by class basis. Being able to integrate any and all possible resources with minimal refactorization is a key challenge facing this project.

The stability of the users' connections to the Digital Backpack service is also something to consider. Students will not always be able to stay connected to the internet while their documents download or upload. The chosen solution needs to be able to handle any sudden interruptions in the connection and have the ability to resume its work with minimal overlap.

2.4.2 Requirements

Team DigiLearn is developing a service that will allow content to be pulled from multiple resources requested by the client, store it in a uniform format so that it can be distributed to users, and displayed in a consistent format across multiple devices. The key features are defined as such:

Uniform Data Storage - Data pulled from resources will be stored in a uniform format to allow for it to be parsed in a consistent manner. Storing it in this way will also allow for data from many resources to be sent to and interpreted by user devices with different operating systems consistently.

Multi-source compatibility - The DigiPack service should pull data from any source and create a file containing all of the relevant information from that source. Resources requested by the client include: Google Search, Google Docs, Google Classroom, Khan Academy, and Youtube.

Scalability - DigiPack will need to be able to make new sources available to users as the reach of the project grows without having to refactor the entire supporting structure. Integrating new resources will take time no matter what format is chosen however; simplicity would make large scale implementations possible.

Stability - Establishing a stable connection with a user's device once an internet connection is available is a key functionality of the DigiPack service. In the event of a disconnect in the middle of operation, the connection should be reestablished once internet connectivity is detected and transfers should resume where they ended.

2.5 Responsible Data Management

2.5.1 Purpose

It is imperative that the Digital Backpack handles sensitive user-data End-users must be confident that any information they transfer using the Digital Backpack is inaccessible by unauthorized users. Furthermore, the Backpack must be compliant with the Family Educational Rights and Privacy Act (FERPA) which makes fastidious security exceedingly important. Given the nature of the proposed network architecture, there are four main points of intrusion to the system: the user's device, the proxy server, the database, and the network connections between these devices. Each of these points must be maximally secured against intrusion to ensure the inviolability of user data.

2.5.2 Requirements

Security will be a consideration for every aspect of the system. For the purposes of the working prototype, the system must have standard security against intrusion. The following items must be implemented for The Digital Backpack to be appropriately secure:

Encryption - Any and all sensitive data stored at any location must be encrypted. Additionally, this data must be encrypted while it is being transported between devices. All encryption keys must be securely handed and stored as to prevent unauthorized access.

Authentication - Communications between devices must be authenticated to confirm that any information served is served only to authorized devices. Additionally, user accounts must use authentication to prevent unauthorized account access.

Device Security - The end devices, the user's device and the proxy server in this case, must be secure against virtual and physical intrusion. Physical protection is primarily in the hands of the use, so The Digital Backpack must keep user information inaccessible to unauthorized individuals even in the case of physical tampering.

Team DigiLearn

3.0 - Technology Analysis

This section goes into depth on the specific technologies that will be used to address the challenges discussed in section 2.0. Each subsection will list the potential alternatives, the chosen approach, and the feasibility of the technology.

3.1 User Interface

The User Interface is an essential part of the project as it would be the main section that the users will interact with. The application will be required to have various designs and prototypes that will be showcased to the client and users. The tools to overcome these needs would be researched to help deliver the project.

3.1.1 Alternatives

In deciding what tool to use for designing the UI, some of the tools options were:

Sketch - Sketch is a digital design toolkit which is based on vector graphics. It's a great tool to use when designing the interface, but it is only based on iOS. Cross-platform software development is not supported. There is also an account membership fee needed.

InVision Studio - InVision Studio is a free vector-based digital product design platform that helps develop screen design processes. The main idea is to make it easier for UI designers to create prototypes of the designs they envisioned. The downside of this is that it doesn't offer the designing aspect. It would be a great tool to use when we're creating prototypes for the client.

Axure - Axure is similar to InVision, as it's also a digital product design platform. However, it is not as effective as InVision because it tends to be more glitchy when trying to design multiple screens, and it can get laggy when switching from different screen designs. There is an account membership fee required to be able to use the tool.

Adobe Experience Design - Adobe Experience Design is a relatively new tool that allows the combination of Sketch and InVision Studio because it provides the designing process and prototyping mobile applications. It encourages collaborations within multiple projects.

3.1.2 Chosen Approach

The following table is the summary of the result of each tool based on the criteria:

Tools	Easy To Use	Collaboration	Compatibility	Total
Sketch	**1**	1	0	2
InVision Studio	1	**1**	1	3
Axure	0.5	0.5	**1**	2
Adobe Experience Design	1	1	**1**	3

Table 3.1: User Interface design tool comparison chart | Rating Scale: 1-3, 3 = best

In conclusion, the application will create designs and prototypes using the Adobe Experience Design to tackle the application interface challenges. It satisfies the criteria needed to fulfill. As shown in the comparison chart, The Adobe Experience Design did tie with the InVision Studio. However, it is much more beneficial to use Adobe Experience Design as it allows to make design sketches and quickly create wireframe prototypes of the design completed. Rather than InVision Studio, which only focuses on producing prototypes. Since the Adobe Experience Design is relatively new, InVisionStudio will be used as a backup.

3.1.3 Feasibility

To prove that Adobe Experience Design will be able to help us in the design process of the project, we plan to design a simple UI that will demonstrate the following process:

- User login layout
- Homepage layout
- User profile layout

3.2 Cross Platform Development

The cross platform development tool is important to the Digital Backpack application. The tool needs to be able to create a simple and easy-to-use application that is able to work with IOS, Android, and Web Applications. This tool also must have an architecture that is visibly pleasing and runs fast and without errors to make it easier and more enjoyable for the customers. The framework that the application is coded with must also be effective for debugging. The following section includes the tools that were looked at and researched, as well as the tool that has been selected to use for cross-platform development for the user application.

3.2.1 Alternatives

Flutter - Flutter is a service recommended by the client Dr.Vigil-Hayes. Flutter is a cross platform tool made by Google. Flutter works with Android, IOS, and web devs. Flutter enjoys priding itself on working very quickly allowing creators to build UI and fix bugs within seconds. Using Google's code language, Dart, Flutter creates beautiful and simple looking applications with a large selection of widgets.

React Native - While searching forums of cross platform development compared to Flutter, I found React Native. React Native is a cross platform tool that can be used with both Android and IOS. React Native was released in 2015 by Facebook and is used by that platform. It is also used with many other apps such as Instagram, Discord, and Skype. Using JavaScript, React Native allows its users to create a simple UI that allows you to integrate quickly.

Ionic - While searching for other tools similar, but providing differences, to Flutter and React Native, I came across Ionic. Ionic is a UI Framework that is flexible across many platforms. The technology was created in 2012 and is used by companies such as AAA and NASA. Ionic uses JavaScript to easily create fast, simple, interactive applications. The cross platform Ionic creates allows the architecture the user has created to adapt to different platforms. Other than JavaScript, Ionic grants its users the ability to use other frameworks such as Angular, React, and Vue.

3.2.2 Chosen Approach

The following is a careful consideration of each tool and what services they would provide for the project. Flutter was the chosen cross platform development tool created by Google. There are various reasons why this technology was chosen. The design that Flutter provides looks very nice and neat and the application runs very smoothly, which is an ideal experience for the users of the Digital Backpack. Flutter has a wide variety of widgets to use for the application as well. Although React Native satisfies all requirements, forums comparing it to Flutter shows the cons the tool has. While both have great simple UI, Flutter adapts better to different platforms, while React Native looks the same.

Flutter tends to also run and code faster than React Native. Flutter shares a lot of pros with Ionic as well. Some of the main differences between the two is that Flutter has a lot more options with UI and the performance is not as smooth. The table below shows the ratings for all of the tools and the requirements for the project.

Tools	**Simple Architecture**	**Reaches Across Multiple Platforms**	**Functions Fast and Without Errors**	**Easy to Fix Bugs and Update**	**Total**
Flutter	2	**3**	**3**	**3**	**11**
React Native	2	2	2	**3**	9
Ionic	**3**	**3**	2	2	10

Table 3.2: Cross Platform Development Comparison Chart | Rating Scale 1-3, 3 = Best

The table shows that React Native is easy to fix bugs and update, but is okay and running smoothly, multiple platforms, and simple architecture. Ionic has a nice simple architecture and reaches across multiple platforms, but is mediocre at running smoothly and fixing bugs and updating. Flutter has an architecture that might not be as simple, but reaches across the necessary platforms easily, runs at a good quick and smooth pace, and is easy to fix bugs and update which is why it has been chosen as the cross platform development tool for the Digital Backpack.

3.2.3 Feasibility

To prove the feasibility for Flutter, there must be some testing for the tool first. After downloading the tool, there will need to be research into the Dart language and how to implement it. Flutter also includes multiple widgets that will help with the design of the application. Additionally, experimenting with the widgets will showcase which potential features to include in the application. One of the widgets that Flutter includes is a platform view widget. Once this widget is implemented through the web-based browser, Android, and iOS sides, they can be put together to create a platform view for both.

3.3 Opportunistic Content Delivery Networks

Opportunistic Content Delivery Networks (oCDNs) facilitate asynchronous communication in situations where a consistent connection is not available. Delay Tolerant Networks (DTN) is a term sometimes used interchangeably with oCDNs. This document will refer to the architecture as oCDNs. The following sections discuss the oCDN architecture, web app frameworks, cloud services, the chosen approach, and the feasibility of this technology.

3.3.1 oCDN Architecture

The traditional internet protocol suite is primarily based on TCP and IP. TCP/IP has certain assumptions that do not allow for prolonged delays in communication. The oCDN architecture uses a method of store-and-forward message routing to account for these variable delays. The data moves from a storage place on one node to a storage place on another node. These storage places can hold messages indefinitely, unlike the short-term storage in TCP/IP, which only expects to buffer data for a short period of time.

The oCDN architecture operates under the following assumptions:

- Intermittent Connectivity
- Variable delays
- Asymmetric Data Rates
- High Error Rates

To account for these necessary nodes will be developed for store-and-forward routing within the system. The backend server framework must be able to accommodate this architecture.

3.3.2 Web App Frameworks

A back-end web app framework will be used to develop the oCDN. Web frameworks support the development of web applications and are highly customizable. Members of Team DigiLearn are most familiar with the programming languages Python and Java. The following frameworks are based primarily in these languages.

The desirable characteristics for the server framework are as follows:

- Integrates necessary services such as Google Classroom*
- Has thorough documentation
- Preferred language
- Scalability

**Integration with external sites further discussed in section 3.4*

Google Cloud - Google Cloud is Google's hosting service. As a major corporation, Google promises security and privacy. There is a free trial tier for their cloud services, but pricing varies depending on what specific services are needed.

Azure - Azure is a cloud hosting service that offers broad catering to a wide variety of needs. Webapps can be built using the framework of the developer's desired language of choice. Azure has stability but at higher operating costs.

The best cloud service that suits the needs of this project is DigitalOcean. The explanation for this is further discussed in section 3.3.4

3.3.3 Chosen Approach

Based on recommendations from the client and an analysis of the available services, Django was chosen to serve as the web app framework, and DigitalOcean to serve as the cloud computing service for the project.

Tools	**Language**	**Documentation**	**Overhead**	**Total**
Django	**3**	**3**	**3**	**9**
Flask	**3**	**3**	2	8
CherryPy	**3**	1	2	6
Apache Wicket	2	2	2	6

Table 3.3: Web App Framework Comparison Chart | Rating Scale 1-3, 3 = best

Django is one of the most popular Python web app frameworks. Common features and templates are built in which allows for quick set up and saves development time. These features include code for common tasks such as security, session management, and database manipulation. The application will be handling user data and Django offers high security for managing user accounts and passwords.

Tools	Configurability	Ease of Use	Scalability	Pricing	Total
DigitalOcean	1	**3**	**3**	**3**	**10**
Google Cloud	2	2	**3**	2	9
Azure	**3**	1	**3**	1	8

Table 3.4: Cloud Service Comparison Chart | Rating Scale 1-3, 3 = best

Jango - Django is a Python-based fullstack web application framework that offers many core functionalities from the start. Less third party plugins are needed, allowing developers to get straight to creating their web application. Django is ideal for larger projects, highly scalable, and is backed by an extensive community.

Flask - Flask is a lightweight Python web application framework that is flexible, especially for smaller projects. Some features of Flask include integrated unit testing, RESTful and HTTP request handling, and clean API. Flask has thorough documentation and is backed by an active community

CherryPy - CherryPy is an object-oriented Python web application framework that is lightweight and flexible. This framework gives developers a lot of freedom when it comes to creating project structures and architectures. REST APIs can be created with the built in tools given. CherryPy is not as popular as other frameworks like Django and Flask, so there may be a less active community.

Apache Wicket - Apache Wicket is a Java component oriented web application framework. Component-based web app frameworks, in contrast with action based web app frameworks, use components to display data that can react to events from an end user. A benefit of component-based development is that it ensures UX consistency.

The best proxy framework that suits the needs of this project is Django. The explanation for this is further discussed in section 3.3.4.

3.3.4 Cloud Services

To host the server cloud services will be utilized. Cloud services can be used alternatively to physical data centers, which can be much more costly. These services offer computing power, storage, and databases. Some use cases for cloud computing include data backup and recovery, email, virtual desktops, software testing, and web applications. The benefit of cloud computing is its flexible pricing and scalability.

Desirable characteristics of the cloud software include the following:

- Lightweight
- Easy to use
- Scalability
- Fair pricing

DigitalOcean - DigitalOcean is a cloud hosting service built by developers with a simple interface to provide for easy configuration. Because of its simplicity and ease of use, it is targeted towards smaller-scale projects that don’t require an all-encompassing cloud computing ecosystem.

DigitalOcean suits the project needs as a simple cloud service to host the proxy server. The goal is to make a small scale prototype for testing to work for around 25-50 students. DigitalOcean has the benefit of being an easily configurable service while still having the potential to scale, making it perfect for the envisioned project. Additionally, DigitalOcean offers rates on the cheaper side when compared to cloud services intended for extensive enterprise-level systems.

3.3.5 Feasibility

To test the feasibility of this technology, a simple proxy server will be implemented and hosted on DigitalOcean, using their free starter package. The capabilities will be tested by writing a service that periodically makes calls to 3rd party APIs. A testing service such as Postman will be used to make requests to the server.

3.4 Integrating Multiple Resources

The range in maturity, education, and age of the users presents an interesting technical challenge. Users need to be able to pull resources from multiple services, pull resources provided by teachers, have the option for educators to restrict the content available to their students, all while still having the ability to access just about anything educational through the DigiPack app. End user storage capabilities is also a limiting factor in this scenario, many mobile devices only have approximately 16GB of on-board storage, much of which is taken up by the operating system and applications. Therefore the developers also need to find a solution that is compressible and can possibly be stripped down to the raw content from the connected services.

Internet connection stability is not a given with this project. The DigiPack service also needs to be able to handle sudden disconnections, packet loss, and other forms of connection disruptions. If, for example, a user has to go home immediately after school and does not have time to finish downloading or uploading their files, a note should be made of where the transfers were stopped and when a connection is reestablished, resume the process from there.

The main factors of the above scenario that this section will address are the ability for resources to be pushed and pulled from the services that have been outlined for by the client. These services include:

- Google Classroom
- Google Docs
- Google Search
- Khan Academy

- Youtube

as well as providing an interface that allows for other services to be "connected" as needed. Because the content from each of these services is so diverse the server needs to be able to organize it in such a way that allows for content to be filtered and then pushed to the students devices without disrupting the homogeneity of the app.

At the core of these requirements there are seven characteristics that will be addressed:

1. Support for the requested services.
2. Scalability for future support of other services.
3. Modularity in the event that one of the implemented services changes their content format.
4. The ability to implement a localized structure and/or format to the data received from each service.
5. Simplicity in the data from each service so that it can be scaled up or down depending on the needs of the end user.
6. The ability to handle unstable connections and resume operation after potentially long periods of lost connectivity.
7. Inexpensive/free due to the non-profit nature of this project.

3.4.1 Alternatives

Parsing HTML files/web crawlers

OpenSearchServer https://www.opensearchserver.com/documentation/api_v1/web_crawler.md

The initial idea for collecting web content for users was to implement a web scraper/crawler that would fetch HTML files and embedded objects as they were needed and store the raw data locally. Web crawlers and scrapers have been around for decades and are a tried and true method for pulling web content for local storage. The above link is a web crawler with a relatively simple and easy to use interface.

Network Sockets

Python Socket - https://docs.python.org/3/library/socket.html

To establish a connection with users a simple network socket protocol could be used in tandem with a web crawler to pass them the data pulled from websites. Network sockets have been in use for just as long if not longer than web crawlers and were originally defined by Joel M. Winett in 1971 for RFC 147 (https://tools.ietf.org/html/rfc147). Like web crawlers, sockets

are a simplistic solution that can be relatively easy to implement and allow for multiple connections to the same data at the same time.

RESTful Services

REST architecture - https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

The client, Dr. Vigil-Hayes, recommended REST APIs and associated RESTful services. REST services as a concept were defined by Roy fielding in 2000 alongside the HTTP 1.1 protocol. REST services provide a structure to requesting web content and local standardization of content for ease of use and modularity. While being a relatively new technology, many large organizations and companies have begun to adopt the concept in an attempt to standardize data that is used across multiple interfaces, operating systems, and applications.

3.4.2 Chosen Approach

In combination with the requirements specified by the client and the characteristics defined in section 3.4, each of the options that were introduced in the previous section were analyzed to try and find the best solution for the project. First a few key requirements were defined as more crucial than the others, namely:

- The ability to interface with the services requested by the client.
- Modularity to be able to add interfaces as needed in the future.
- Inexpensive/free to use due to the non-profit nature of the project and limited budget.

These three requirements shaped the initial scope of research into possible alternative solutions. The alternatives listed in section 3.4.1 all fit this description and so the developers felt as though these were a good place to start testing.

While the developers did not do any testing specific to this project of any of these solutions, they pulled from the team's past experience with each of these solutions to come to a decision on which solution to use in the initial implementation. A few members of the team are currently taking CS 460 - Computer Networks with the client at NAU and have experience implementing network sockets, others have professional experience using and implementing RESTful services. Most grew up with pre-Google internet and have experience with web crawler based search engines. All of this experience has allowed them to compile a list of pros and cons to each alternative.

Web Crawlers

Pros - Allow for content to be pulled from any online resource at any point, relatively simple implementation, many different iterations to choose from in the event there is a shortcoming or feature that the chosen solution does not implement.

Cons - Content is slow to produce because each page has to be parsed individually, content is not in a standard format after parsing, large objects or stream data such as video or audio cannot be easily accessed or stored, potential legal issues acquiring and storing audio or video content. This only solves half of the problem, getting the web content. It does not allow for content to be passed to the users on its own.

Network Sockets

Pros - Fast connections, secure connections can be implemented, real-time data transfer is possible, very little overlap in data during connection instability due to sequencing (TCP)

Cons - connection instability could lead to corrupted files on either side of the connection, the required computational overhead for a continuous connection can be inefficient for small scale hardware implementations or large scale software implementations. This as well is also only half a solution, it only allows for a connection to be established with the users, does not inherently pull web content.

RESTful Services

Pros - Interfaces with all of the requested resources and many more, building “interpreters” for each of the requested services allows the developers to build a local standard for data pulled from each source and therefore create a separation in development between the mobile app and the server, the inherent “statelessness” of REST services allows for dynamically requested and provided data both on the server-resource connection and the client-server connection with little to no need for a persistent connection.

Cons - REST services do not inherently have any sort of authentication built into them to ensure that the user is who they say they are, however because of the fluid nature of RESTful services a solution to that can be implemented and integrated into the service.

The following table is a visual representation of the pros and cons listed above in relation to desired characteristics:

Tools	**Support**	**Scalability**	**Modularity**	**Localized Structure**	**Data Simplicity**	**Connection**	**Cost**
Web Crawlers	**3**	1.5	1	2	1	n/a	**3**
Sockets	n/a	n/a	n/a	n/a	**3**	2	**3**
REST Services	**3**	**3**	**3**	**3**	**3**	2	**3**

Table 3.5: Resource Integration Comparison| Scale: 1-3, Best = 3

As shown above, REST services meet almost all of the requirements with flying colors and should allow the developers to build a service that fulfils the needs of this project and allow for modifications to each part of the system independent of the requirements of other functionalities.

3.4.3 Feasibility

Because RESTful services are more of a concept than a fully fleshed out API or library the developers will need to do quite a bit of testing and development to create a service that fits the needs of this project. Initial development and testing plans will be to settle on a format (HTML, XML, JSON) to use as an internal standard and what those files will include. From there development of a storage solution and interface that will have three core sections:

1. A way to request and receive files from the interpreters
2. An interface with the local resource database
3. An interface to send and receive data with the user(s)

These three sections will be further broken up to improve functionality, modularity, and minimize coupling. The interpreters will have to be built for each resource and starting with one resource interface as a proof of concept and expanding as needed. These interpreters will take the data from each resource's existing REST API and create a file based on the previously mentioned local format. An interface with the servers database to store, organize, and search through the RESTful data will also be needed. The developers will need to implement a way for users to push data (documents, emails, etc.) to their respective resources through the app. Finally a way to ensure that the users are who they say they are and a way to tell that through the data sent between their device and the server will have to be created. An interface will be implemented to parse requests and cross reference login data with the servers local authentication specified in section 3.5.

The development team has confirmed that all of the resources that the client requested do have publicly available REST APIs, however some (namely Khan Academy) have legal requirements for their use. The legal issues will be discussed further with the client to ensure that the use of those APIs is within legal bounds. The APIs themselves will have to be tested to get a clear understanding of the content provided by them, the available functionalities, and their structure. Simply due to the massive scale of the company, the Google based services that will be integrated with the Digital Backpack project are well documented. Some of the smaller companies however, will need to be thoroughly unit tested to ensure that the interpreters are able to pull or push their content as needed. The development team plans to create a rough framework for the eventual interpreters for each resource to be able to take a look at the returned data and structure more thoroughly.

Below is a flow chart generalizing the use cases and interfaces that the team currently foresees the REST service needing to be fully functional. The green represents services or

interfaces that already exist, yellow for the DigiLearn serve as a whole, orange for the REST service, red for the internal functionality of the REST service, purple for the authentication covered in section 3.5 and blue for the oCDN connectivity functionality of the server.

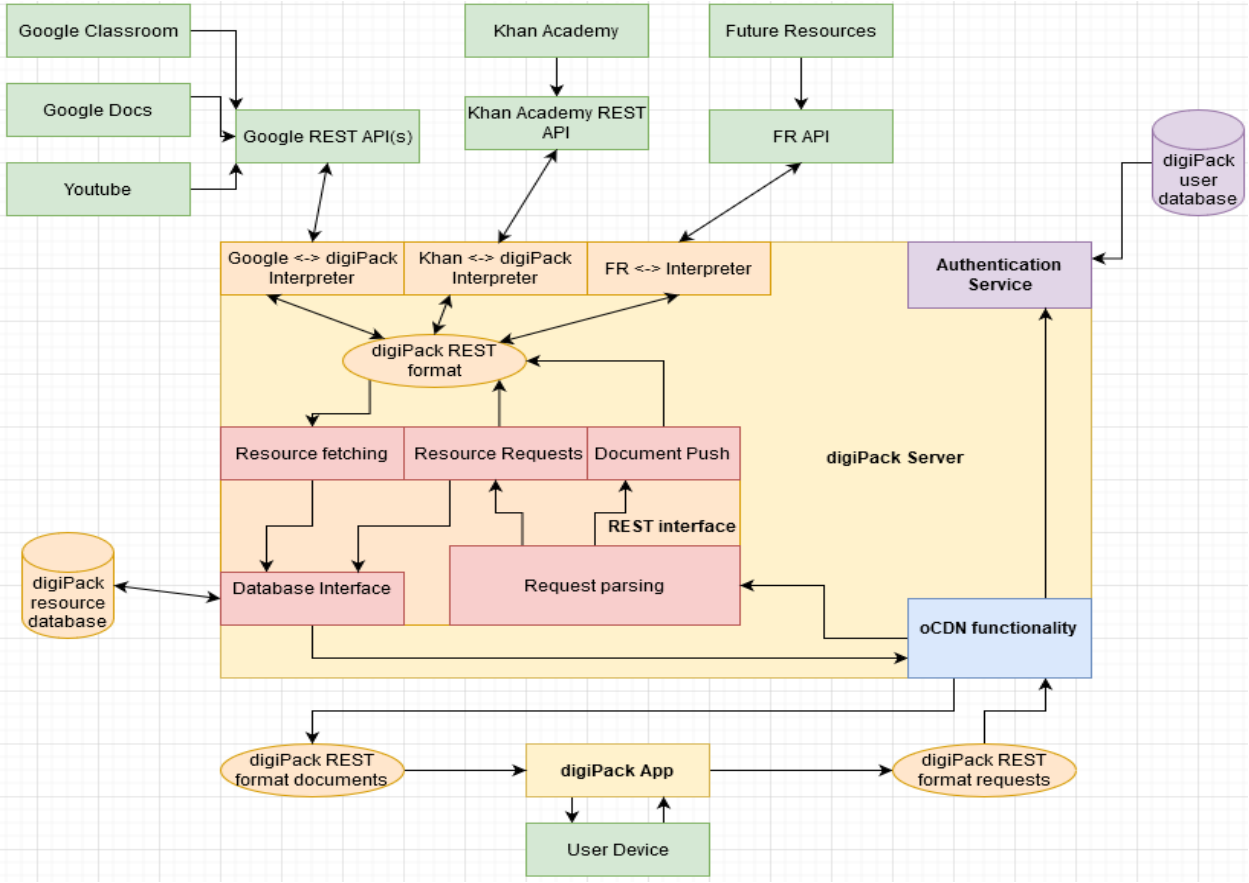

Figure 3.1: DigiPack Server rough architecture flow chart

3.5 Security

The security of The Digital Backpack is integral to the product being usable. There are four main components of the system which must be secure: the proxy server, the user's device, the database, and the network connections between these devices. The following sections will explore the possible approaches for each of these components, and this section of the document will conclude with a summary of the approach to system security and the feasibility of keeping user data secure in this context. To outline the specifications for the security of the DigiPack system, consider the following desired characteristics:

Proxy Server

- The proxy server must only accept requests from the user's device and from authorized external services.

- The proxy server must be protected against intrusion from unauthorized sources.
- In the case of intrusion, user data stored locally must be further secured with strong encryption.

User Device

- The application on the user device must only communicate with the proxy server. It must be impossible for the application to communicate with any other device.
- Other applications on the user device must be prevented from subverting this connection to communicate with the proxy server.
- In the case of intrusion, user data stored locally must be further secured with strong encryption

Network

- Connections to every device in the system must only be accepted from authorized sources. These connections must be protected from interception.
- Sensitive user data transported between devices over the internet must similarly be safe from tampering or theft.

3.5.1 Proxy Server Security

The proxy server is acting as an intermediary between the user and a variety of services on the internet. As such, it is unavoidable that the proxy server will have to store sensitive information while the user's device is offline. As a matter of best practice, it is essential that user data is secure for the entirety of its life on the proxy server. The following technologies will be considered for this purpose:

Server Firewall

The proxy server could be equipped with a firewall, such as the popular Uncomplicated Firewall (UFW) to manage traffic that comes into the server. This would be an effective way, in conjunction with OAuth2, to prevent the server from being compromised.

Encryption

As an extra layer of protection, any user data that is stored on the server should be encrypted so that it is still secure in the case that the server is compromised. Standard RSA / AES encryption would suffice for this purpose, but this raises the issue of keeping essential keys safe. One option is to store the keys in memory once the server is started. This would require manual intervention, however, and may result in encrypted data being inaccessible in the case that the server must be restarted.

3.5.2 User Device Security

The Digital Backpack must function offline. Thus, the Digital Backpack must store potentially sensitive data locally. This makes the user device a potential place where user data may be compromised. As a result, it is essential that all data on the user's device is stored securely. The following technologies will be considered for this purpose:

Secure Storage

Mobile devices offer a secure storage environment (Keychain for iOS and KeyStore for Android) explicitly for the storage of particularly sensitive data. This is a small amount of storage, however, and so would only be valuable for encryption keys and passwords. In addition, a browser application would not be able to rely on this service.

Encryption

User data can be encrypted when stored locally. The encryption keys could then be stored in the secure storage box described above. The main drawback is that encrypting and decrypting data can be a time-consuming process.

User Sessions

Appropriately managing user sessions will increase the security of the user's device and thus the system as a whole. User session should be terminated periodically, a login should be required. Biometric authentication will not be permitted as this technology may exclude many devices and limit the scope of the potential user base.

External Authentication

If necessary, user sessions could be authenticated by matching the inputted password with a cryptographically hashed password stored in a separate database. This solution requires a connection to the internet and thus is not ideal for most applications in the context of the Digital Backpack.

3.5.3 Network Connections

Any communication occurring between devices must be secured to prevent requests from being tampered with and to keep user data secure during transit. In the following list, multiple technological alternatives are detailed:

Virtual Private Network (VPN) Service

The use of a proprietary VPN service would ensure that any user data sent over public networks, as is often done in the context of the Digital Backpack, is protected until said data reaches the VPN's endpoint. However, user data would then be vulnerable in transit from the VPN's endpoint to the end device. In addition, the use of this outside service would require a

monthly fee which may be prohibitive to the deployment of the product.

RSA / AES Encryption

A combination of AES and RSA encryption would allow for all user data and requests to be encrypted for the entirety of their transit from one endpoint to another. The public-key system in RSA will enable the automation of trading keys and establishing a secure connection of encrypted data. This solution raises the issue of securely storing these encryption keys. Additionally, encryption unavoidably has an impact on performance, but the bulk of this performance impact can be handled offline, so it won't interfere with the limited, opportunistic connections to the internet.

OAuth2

OAuth2 is an industry-standard authentication framework which would resolve the issue of making sure all end devices in the system only communicate with authorized devices. OAuth2, however, does nothing to protect data while it is being transferred.

Transport Layer Security

Transport Layer Security (TLS) is a standard security measure and part of the https protocol. TLS implements a secure, encrypted, and bound connection between the client and the host. This protocol is supported by Flutter libraries, which would make implementation relatively simple. On the other hand, the fact that this protocol is standardized may be a vulnerability.

3.5.4 Chosen Approach

After analyzing the options for security presented above, the following analysis was compiled to summarize the efficacy of the various available technologies. Security will be addressed by category of the three points of vulnerability outlined in Sections 3.5.1 through 3.5.3.

Proxy Server

The proxy server must store all user data in a way that prevents unauthorized access. Additionally, the proxy server must only respond to requests from authorized devices. Finally, the proxy server must be able to communicate in a way that is compatible with external resources such as the ones described in Section 3.4. Consider the following analysis of the technologies discussed in Section 3.5.1:

Tools	Secure Storage of User Data	Authentication with Digital Backpack	Authentication with External Resources
Uncomplicated Firewall	0	**3**	**3**
Encryption	**3**	0	0
OAuth2	0	**3**	**2**

Table 3.6: Server Security Comparison Chart | Rating 1-3, 3 = Best

Due to the variety of needs of the proxy server, all of the above technologies will be utilized. All user data that is stored on the proxy server will be encrypted with a local AES encryption key that will be stored in the server's RAM to prevent unauthorized access. The Uncomplicated Firewall will be used to secure the server against unauthorized access. OAuth2 tokens will also be used when the server is communicating with compatible devices, such as The Digital Backpack app and select external resources.

User Device

The user device must manage user sessions and provide authentication to prevent unauthorized access to any user data. In the case of unauthorized access, all user data must be encrypted while stored as an extra layer of protection. Encryption keys and passwords need to similarly be kept away from unauthorized access. All of these functions must be performed while the user device is offline. Consider the following analysis of the technologies discussed in Section 3.5.2:

Tools	**Session Management**	**Encryption**	**Key Storage**	**Offline Functionality**
Secure Storage	0	0	**3**	**3**
Encryption	0	**3**	0	**3**
Login—Internal	**3**	0	0	**3**
Login—External	**3**	0	0	0

Table 3.7: User Device Security Comparison Chart | Rating Scale 1-3, 3 = best

Based on the analysis above, the following solution will be implemented. Access on the user device will be managed with user sessions that are authenticated with a username and password. The password will be compared to a hashed password stored locally in the device's secure storage. Sensitive information stored on the user device will be encrypted with an AES key. This key will also be stored in the device's secure storage.

This solution will be slightly different in the case of the web application. The web application will authenticate the given password to a hashed password stored externally in a user database. No user data will be stored locally on the user device and will be instead served directly from the resource database.

Network

Network connections must all be authenticated to prevent unauthorized access to the user's device or to the proxy server. In the case that the data is intercepted, all sensitive information must be securely encrypted as an extra layer of protection. All of these operations must be performed in a way that does not significantly impact transmission speed when the user device

device is opportunistically communicating with the proxy server.

Tools	Connection Authentication	Data Security	Impact on Transmission
VPN Service	0	2	0
RSA / AES Encryption	0	**3**	**3**
OAuth2	**3**	0	**3**
TLS	0	**3**	**3**

Table 3.8: Network Security Comparison Chart | Rating Scale 1-3, 3 = best

Note: *A 3 denominates a low impact on transmission speed.*

Based on this analysis, the following solution will be implemented. All network traffic in the Digital Backpack system will be performed in accordance with the https protocol utilizing TLS for transmission security. Connections will be authenticated using OAuth2 technology, as it is the industry standard.

3.5.5 Feasibility

There are three main concepts that need to be tested to prove that this approach is feasible. The following list will enumerate these concepts as well as how they will be tested:

Encryption

Encryption will be tested by developing a small application in Flutter and for the proxy server that can successfully encrypt and decrypt test data while keeping encryption keys stored in a fashion that is inaccessible to users.

OAuth2

OAuth2 will be tested by developing a small application in Flutter and for the Proxy server that can exchange authentication tokens and successfully use said tokens to authenticate connections.

TLS Protocol

The TLS protocol will be tested by developing a small application in Flutter and for the proxy server that can successfully transmit data in a way that is compliant with TLS protocol.

Team DigiLearn

4.0 - Technology Integration

This section discusses the final plan for the system. Each aforementioned technology comes together to create the Digital Backpack. To begin, please consider the following preliminary system diagrams which summarize the proposed architecture for the Digital Backpack system. In these diagrams, yellow is used to encapsulate entire parts of the system. Purple denotes functions that interface with databases. Blue denotes parts of the system that relate to the oCDN functionality of the system. Orange is used to denote miscellaneous functions performed by the Digital Backpack.

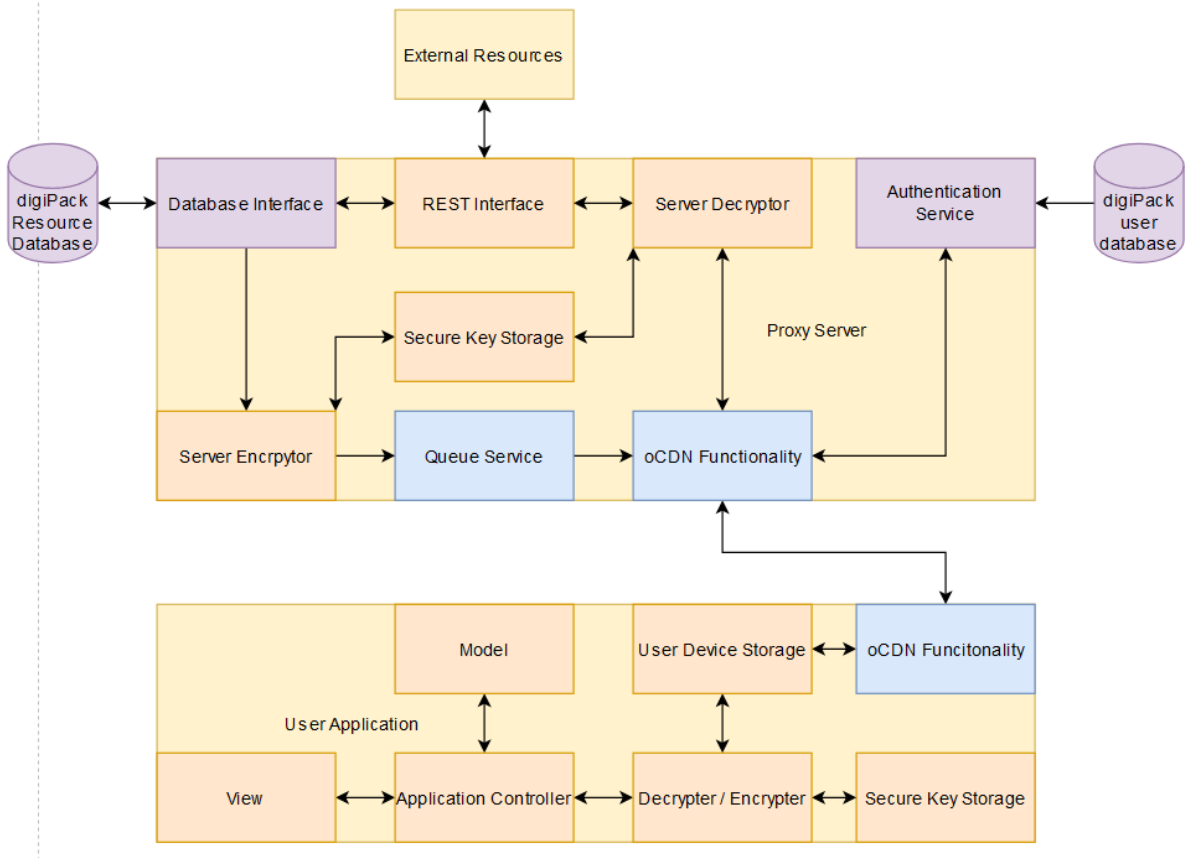

Figure 4.1: DigiPack Architecture System Diagram

4.1 The Proxy Server

The proxy server acts as an interface between the DigiPack and the internet. To this end, the Proxy server includes a REST Interface that standardizes resource formatting between various external services, such as Google Classroom, and the Digital Backpack system. The proxy server also communicates with the DigiPack resource database and the DigiPack user database for the purposes of resource storage and authentication. To enable efficient transmission between the proxy server and the DigiPack, some data is queued and stored locally so that said resources are ready to be delivered opportunistically as soon as a connection to the user device is established. The proxy server also handles encrypting local data for storage and decrypting data coming in from the DigiPack. In this case, encryption keys are stored in active memory and never written.

4.2 The User Application

The user application serves two primary functions. Firstly, the DigiPack must interact opportunistically with the proxy server. To this end, the DigiPack has oCDN functionality that will detect when a connection is available and autonomously begin communicating with the proxy server. Secondly, the DigiPack must provide a user interface. The application will use a standard view-model-controller design. In this case, multiple views will be available to suit the user's needs. For example, one view will be suitable for child users while another will be designed for use by teenaged users. Like the proxy server, the user application will handle the encryption and decryption of data. In this case, encryption keys will be stored in the device's secure storage environment.

4.3 The Web Application

The nature of the web application demands that it's architecture is slightly different than the architecture described for the user application above. Consider the following system diagram that illustrates this separate architecture.

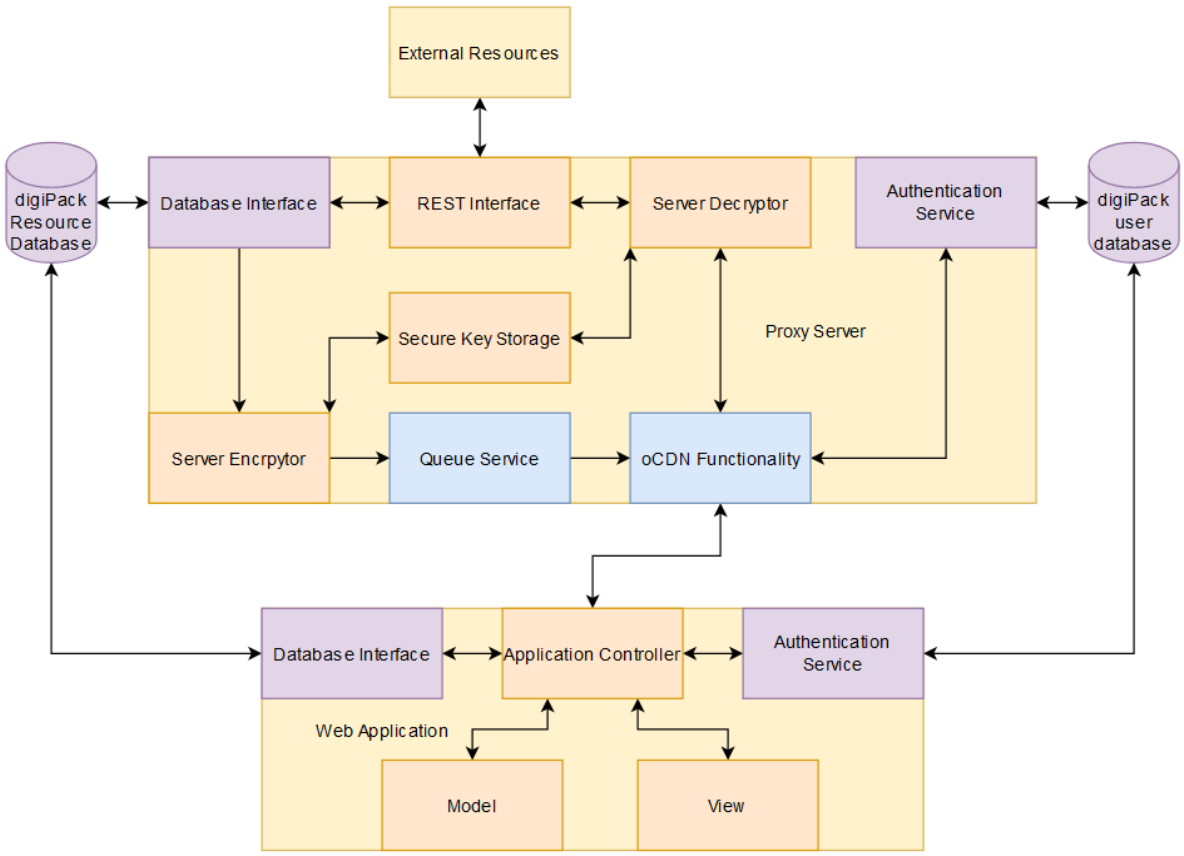

Figure 4.1: DigiPack Architecture System Diagram

The web application will not behave opportunistically. Instead, it will interface directly with the DigiPack resource database to retrieve resources. This will also allow the web application to function without the need to store any user data locally, which is unfeasible in terms of security. The web application will still communicate with the proxy server to deliver requests, receive queued resources, and otherwise behave as a seamless experience between it and the mobile application. The web application will authenticate the use by comparing a supplied password to a hashed password stored in the DigiPack user database.

5.0 Conclusion

Many students across the country lack internet access at home and are potentially struggling to find reliable internet access in general. With the rising prevalence of remote learning, the digital gap and its impact on student outcomes is more prominent than ever. The Digital Backpack offers a solution by implementing an opportunistic Content Delivery Network or oCDN to enable the user application, the DigiPack, seamless transition between online and offline learning and thereby lessen the effects of this digital gap.

This document has enumerated the technological challenges faced by this goal and explored a myriad of potential solutions. For the purpose of making this application available to as many students as possible, Flutter was selected as a platform to simultaneously develop the application for iOS, Android, and as a web application. The user interface for the mobile application will be developed using Adobe Experience Design. For the scope of this project, the Django webapp framework will be used to implement the oCDN architecture. The application will be hosted using Digital Ocean's cloud service. Additionally, a RESTful service will be used to facilitate communication between the system and various external APIs For security, a suite of standard measures will ensure that user data is kept completely safe and secure.

In conclusion, the research into the technological feasibility of the Digital Backpack has established a strong foundation for the overall architecture of the system, concluding that there are no technological challenges that will prevent the creation of the Digital Backpack system.