

# Product Vision and Architecture Document

CarefulAI: ImagineThis V2

## Version history

Version	Date	Authors	Comments
1.0	5 <sup>th</sup> January 2020	L. Cerny, R. Zhou, K. Singh, D. Hossain, Y. Zeng, Z. Li	

# 1 Purpose and scope of this document

This document is created for the NHS, our client, to provide a clear picture of the project from multiple perspectives. Written in business language, it contains detailed information about business goals, constraints, stakeholders, features and their requirements, system architecture and finally the actual development plan. It serves as an articulated vision of what the system will be after we implement it during the 2<sup>nd</sup> term and how we actually plan to develop it.

Our team had been meeting three times a week throughout the 1<sup>st</sup> term as well as working asynchronously in order to create this document. We had thirty-minute meetings with our teaching assistant on Tuesdays, where we showed him our progress. We had internal meetings every Wednesday where we discussed comments on each other's work and worked on the document together. Finally, we had thirty-minute meetings with our client on Thursdays, where we discussed his opinions about the system, requirements, goals and background. By combining needs of the client, an investigation of the existing system and reading the previous development team's report, we figured out the product vision, requirements and architecture to be used.

## 2 Product Vision and Requirements

### 2.1 Background

Currently, the NHS is facing a pressing problem. It takes too much time to develop a new mobile application and ship it to the market. The client reports that apps can take from 6 months to 2 years to develop, depending on their complexity. To shorten the development time, the NHS commissioned a piece of software to be made by UCL students. It is called ImagineThis. Version 1 (V1) was already developed by a group of UCL students during the summer of 2020.

*The primary business goal for the NHS is to shorten development time of applications from 6 months to 1 month.*

The goal of this system is to shorten the development time of application from 6 months to 1 month. ImagineThis achieves that by enabling the NHS to automatically convert projects from a commercial platform called Figma into a functional mobile native code. Figma is a non-coding tool where users create interactive UI designs for mobile applications as proof of concepts [1]. This is used by app designers at the NHS. And since the NHS has an insufficient capacity of developers, automatically generating functional and high-quality code is of enormous benefit. ImagineThis requires minimal effort in coding and task can be done within the order of minutes.

To convert a Figma project through ImagineThis, users are required to authenticate themselves and provide the Figma ID of the project they want to convert. Authentication can occur in two ways: with a token or through OAuth 2.0 protocol. Then the user can input their selected Figma project ID and begin the conversion process. Once that succeeds, he can download zipped source files for the app.

ImagineThis V1 has been deployed and is publicly available at: <http://88.80.186.99/>. Screenshot of the UI is in Figure 1 below. From within the website, it is also possible to download a distribution for local deployment of ImagineThis system.

As well as being used by the NHS, ImagineThis is also used by UCL students at hackathons to cut down their development times.

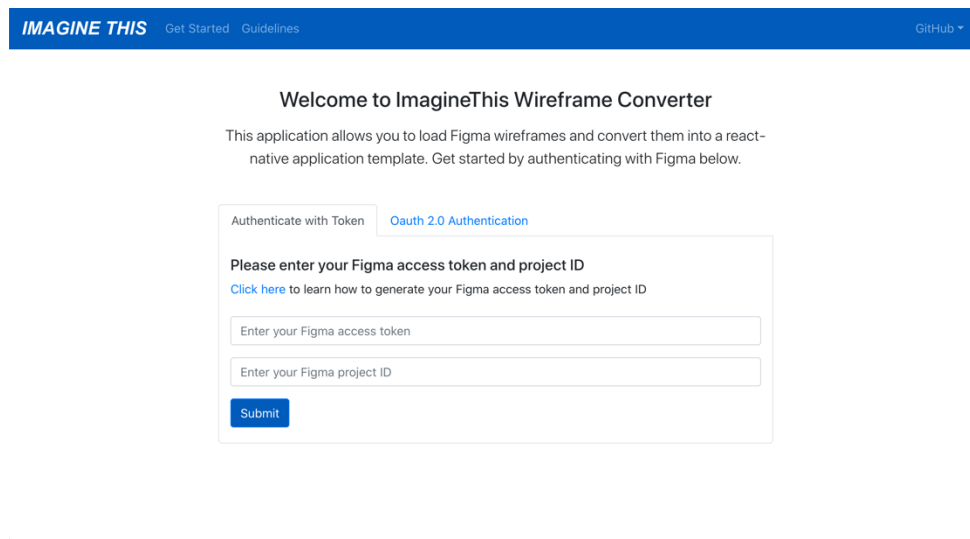
The image shows a web browser window with the 'IMAGINE THIS' logo in the top left corner. The navigation bar includes links for 'Get Started' and 'Guidelines', and a 'GitHub' link on the right. The main heading is 'Welcome to ImagineThis Wireframe Converter'. Below this, a paragraph states: 'This application allows you to load Figma wireframes and convert them into a react-native application template. Get started by authenticating with Figma below.' The authentication section has two tabs: 'Authenticate with Token' (selected) and 'OAuth 2.0 Authentication'. The 'Authenticate with Token' tab contains the text 'Please enter your Figma access token and project ID' and a link 'Click here to learn how to generate your Figma access token and project ID'. There are two input fields: 'Enter your Figma access token' and 'Enter your Figma project ID'. A blue 'Submit' button is at the bottom of the form.

Figure 1: Already functional and existing website of ImagineThis V1.

## 2.2 Business goals

The NHS's main business goal is to shorten the development time of application. This has been achieved with Version 1 of this system, but we aim to improve on it. To do this, we identified several opportunities for expansion - this consists of 2 parts:

- a) adding new functionalities
- b) improving the existing system

For the key new functionality to be added, we focus on enabling end users to test app prototypes and provide valuable feedback to designers quickly. Our client liaison admits that the later the flaws in prototypes are found, the greater the delay in the development cycle. Hence, he wants to be able to “break the prototype as soon as possible”, so that designers can fix those flaws quicker. The new functionality to be added is a feedback system. With this, designers will be able to easily collect feedback from end users that are testing the prototype.

On the other hand, improvements to the current system consist of several minor tasks that will further help in shortening the application development cycle and are not implemented in the current system.

Although the main goal is to shorten the development cycle, we identified system's sustainability as another parallel goal that is key for the success of ImagineThis project. It is crucial that other students or developer groups can easily continue with the system's development once our project finishes. We outline several features that address this goal in section 2.6.

## 2.3 Project constraints

There are several constraints we need to consider in this project.

- **Version 1 already exists:** We are expanding on ImagineThis V1. Hence, many architecture and implementation decisions had been already made and we must obey those to a large extent. For example, Figma, a commercial prototyping platform, was chosen for prototyping. Authors of ImagineThis V1 presented many ways how the app could be built from a prototype: using image recognition or speech recognition and recording of people's creation and interaction

with a Figma prototype. But in the end, they chose to use JSON data from Figma API. Figma and its interactions are the biggest external dependency the system has and must obey.

- **Minimal number of external dependencies:** Client specifically requires as little external dependencies as possible, so we should not add many new ones.
- **Infrastructure on Linode:** The current infrastructure runs in Linode [2], a cloud provider, and the client wants it to stay this way. Although the client did not specify any budget constraints, we need to continue with this infrastructure setup.
- **Open-source system:** The source code and documentation for ImagineThis must be publicly available online. It's a key requirement made by the client, as it simplifies maintainability and future development. And it allows other people to use and develop the system.
- **Time:** Our system needs to be implemented within the 2<sup>nd</sup> term, a period of 12 weeks, which may affect the number of our deliverables and potentially quality of results. Also, since we are group of students working part-time, there is limited time we can dedicate each week.

## 2.4 Stakeholders

Table 1 below describes all stakeholders in ImagineThis project.

Stakeholder	Type	Description
NHS	Client	It's imperative to them that this system is in place so they can cut down application development time, and therefore actualise ideas in a shorter time period.
Designers at NHS	User	People that design the UI for applications and are already trained in Figma. Those can even be clinicians or members of the public. Their concern is to transform their prototypes into a working application and improve their prototypes.
End users	User	These are individuals that give feedback on the created application. Can include NHS designers, developers, testers or even potential end users.
UCL student group	Developer	Group of UCL students responsible for the development of ImagineThis system and its deployment. Their main concern is meeting the client's needs.
Developers at NHS	Functional Beneficiary	People that will be working with the code that the system produces. They want the system to produce a working and high-quality code they can easily work with.
Regulators	Regulator	Since we will store and operate with data, we may need to consider general regulations like GDPR [3], or healthcare industry specific regulations FHIR [4] or openEHR [5].

Table 1: Stakeholders in ImagineThis project.

## 2.5 Project context

This section illustrates the context of our project through context and conceptual diagrams. The context diagram in Figure 2 describes the various entities involved and their interactions. Figma is the commercial platform, ImagineThis our system and Prototype is the generated code of application produced by ImagineThis. The code runs on end users' mobile phones. As already described in the previous section 2.4, there are two user stakeholders: designers and end users.

Furthermore, within the context diagram, there are several concepts that the conceptual diagram in Figure 3 clarifies. Our vision of the overall process of using ImagineThis V2 is the following:

- 1) Designers create a *Project* in Figma.

- 2) This *Project* is then used by ImagineThis to generate code for a given application *Prototype*. This prototype runs on end users' phones.
- 3) End users, when testing it, can submit their *Feedback*.
- 4) Finally, end users can also *Vote* on each other's *Feedback*.

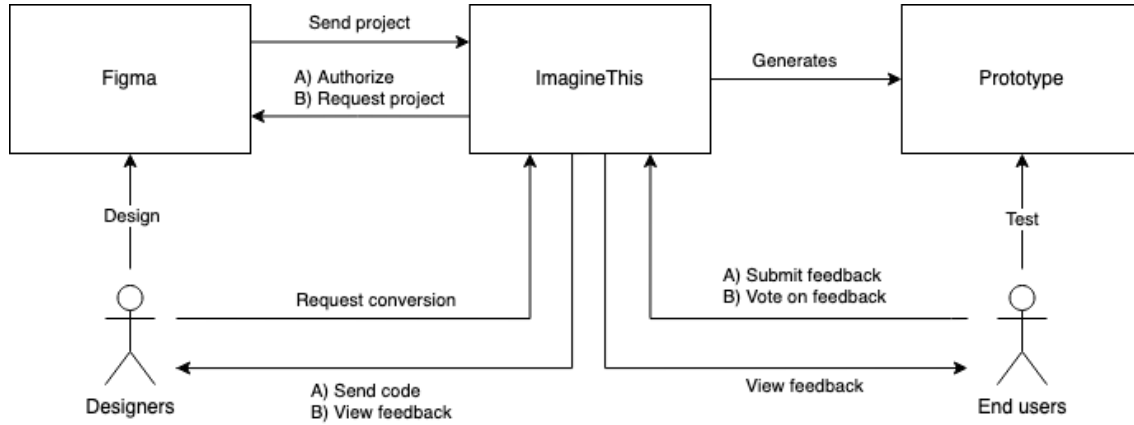


Figure 2: Context diagram of ImagineThis V2 project.

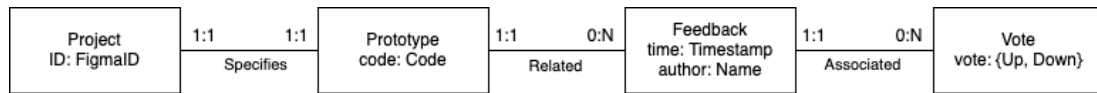


Figure 3: Conceptual diagram for ImagineThis V2.

## 2.6 Features

Figure 4 below is the Goal-Feature map of our system. It shows the features we want to create. Green colour denotes goals and subgoals, whereas the blue colour denotes features that contribute to particular goals. As already explained in section 2.2 there are 2 high-level business goals a) to shorten development cycles, and b) to make ImagineThis a sustainable system. One of the subgoals, *Provide Feedback on Prototype*, is about implementing the new feedback system. Feature contributing to the second subgoal, *Quicker Code Generation for Prototype*, are improvements of the existing system. All features are described individually in the following chapters.

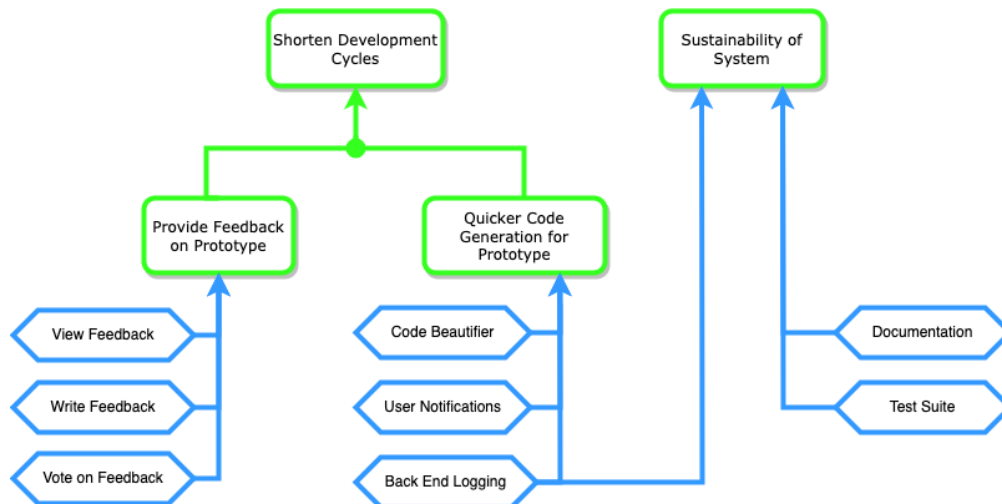


Figure 4: Goal-Feature map for ImagineThis V2 system.

Note that some features are already partially implemented in Version 1. Those are mentioned because we want to improve them.

## 2.7 Feature requirements

This section describes individual features presented in Figure 4 and specifies their requirements. Where applicable, it uses Gherkin language Given-When-Then to demonstrate behavioural requirements of features and these scenarios can be then used for acceptance testing.

### 2.7.1 Feature: View Feedback

**Description:** Users are able to see the feedback written by other users for particular projects. This enables designers to get to know their design flaws or opinions and comments made by the end users. It is a key feature enabling the NHS to create high-quality prototypes.

**Goals:** Provide Feedback on Prototype

**Requirements:**

- Anyone with project ID can view feedback on that project.
- Project had to be converted at least once.

**Rule:** Anyone with project ID can view feedback.

**Scenario:** Designer wants to view feedback.

*Given* a designer has created Figma project with ID LgWqYTZMIjG26oA1CxbWpE

*And* has used ImagineThis to convert it

*When* an end user provides feedback on it

*Then* anyone knowing the ID LgWqYTZMIjG26oA1CxbWpE can view the feedback.

**Rule:** Project had to be converted at least once.

**Scenario:** End user tried to view feedback on a project that has not been converted.

*Given* that a project has not been converted

*When* an end user tries to access feedback for this unconverted project

*Then* the user will see a notification saying it is not possible to view feedback on a project that has not been converted.

### 2.7.2 Feature: Write Feedback

**Description:** End users are able to write their feedback on prototypes. This feature is the key counterpart to *View Feedback* feature described in section 2.7.1.

**Goals:** Provide Feedback on Prototype

**Requirements:**

- End users can add feedback to the prototype they have access to.
- Feedback also contains datetime and author's name.
- Project had to be converted using ImagineThis at least once.

**Rule:** Feedback also contains datetime and author's name.

**Scenario:** End user provides feedback.

*Given* an end user <Name> has access to a prototype with project ID <ID>

*When* <Name> submits feedback "<Feedback>" for the prototype at <Time>

*Then* anyone knowing <ID> can view that <Name> wrote "<Feedback>" on <Time>.

<Name>	<ID>	<Time>	<Feedback>
John Thomson	YpBnZ4aEB2YgGpiOQfxQCU	14 <sup>th</sup> Dec 2020 8:45AM	This is app looks good.
Jane Turner	LgWqYTZMIjG26oA1CxbWpE	21 <sup>st</sup> Oct 2018 12:02PM	Good job!

**Rule:** Project had to be converted using ImagineThis at least once.

**Scenario:** End user tried to submit feedback on a project that has not been converted.

*Given* that a project has not been converted

*When* an end user tries to submit feedback on this unconverted project

*Then* the user will see a notification saying it's not possible to submit feedback on a project that has not been converted.

### 2.7.3 Feature: Vote on feedback

**Description:** End users are able to react on posted feedback, either by upvoting or downvoting them. This can be used to show designers which problems end users think are important, or conversely unimportant or irrelevant.

**Goal:** Provide Feedback on Prototype

**Requirements:**

- End user can react to feedbacks, either by upvoting or downvoting them.
- Feedback is sorted in order of votes (high to low).
- Feedback starts with a vote count number of zero.
- Feedback vote count is increased/decreased by one.

**Rule:** Feedback vote count must be increased/decreased by one.

**Scenario:** End user upvotes/downvotes a feedback.

*Given* that there is an existing feedback with <Before score> vote score

*When* an end user <Action> the feedback

*Then* the feedback's vote score changes to <After score>.

<Action>	<Before score>	<After score>
Upvotes	4	5
Downvotes	4	3
Downvotes	0	-1

### 2.7.4 Feature: Code Beautifier

**Description:** Converted code follows standard styling conventions. Currently, the outputted code has inconsistent indentation. Although developers can use code beautifiers included in most IDEs, it should be a quick fix for us that would simplify developer's life and we would not have to rely on the assumption that they do it themselves. The better the quality of the code ImagineThis produces, the more time it saves for developers.

**Goal:** Quicker Code Generation for Prototype

**Requirements:**

- Converted code follows standard styling conventions.

### 2.7.5 Feature: User Notifications

**Description:** Notify the user about the status of conversion on the front end. This improves the user experience and transparency.

**Goal:** Quicker Code Generation for Prototype

**Requirements:**

- Users receive notification about the conversion status.

**Rule:** Users receive notification about the conversion status.

**Scenario:** Failure in conversion process due to error in Figma project.

*Given* a user has started a conversion of a Figma project

*When* an error occurs during the conversion process

*Then* the user receives a notification onscreen saying which frameworks failed to convert and why.

**Rule:** Users receive notification about the conversion status.

**Scenario:** Conversion operation succeeds.

*Given* a user has started a conversion of Figma project.

*When* the back end successfully generates the corresponding code for the prototype

*Then* the user will receive a notification onscreen informing him that the conversion has been successfully completed.

### 2.7.6 Feature: Back End Logging

**Description:** Back end generates a log file in order to see any issues occurring in conversion due to bugs in ImagineThis. Both of these parts increase the transparency of the system, for users, and for developers and maintainers of ImagineThis, respectively.

**Goals:** Sustainability of System, Quicker Code Generation for Prototype

**Requirements:**

- Back end produces log file which contains the operation log.

### 2.7.7 Feature: Documentation

**Description:** Stakeholders will be able to access all ImagineThis documentation publicly online. This is already implemented, nevertheless, documentation is partitioned into multiple places. One source contains the user guide. Development guide is elsewhere. We want to unite the documentation into one place, so that it is more clear and also easier to maintain.

**Goal:** Sustainability of System

**Requirements:**

- Stakeholder can access all parts of documentation publicly online.
- Documentation includes:
  - User manual on how to use ImagineThis
  - Deployment manual for setting up ImagineThis
  - Development manual for developers of ImagineThis

### 2.7.8 Feature: Test Suite

**Description:** The quality of the current system's test suite is very poor, having statement and mutation coverages both at 5% as measured with PIT [6]. We want to improve this. Specifically, to increase



statement coverage of back end's test suite to above 80%, which is an industry standard for high-quality test suite [7]. This will benefit future development of ImagineThis by making sure that changes do not break existing requirements.

**Goal:** Sustainability of System

**Requirements:**

- Test suite for ImagineThis back end has statement coverage rate of at least 80%.

## 3 Product Architecture

### 3.1 Layered architecture

After evaluating the goals of the project and the features that need to be implemented, we decided to use a well-known **3-tier layered architecture** as it is the most suitable architecture for our project. It is illustrated in Figure 5.

The front-end layer is the ImagineThis website that is mainly used by the users to convert the Figma project to a working prototype with source code and to manage feedback. The back-end layer is the core layer of the system that is responsible for the actual conversion of the source code, returning the output to the users via front-end layer, and managing feedback by storing feedback in the database and retrieving the feedback from the database. The database layer is the actual database that stores the information of the feedback including feedback's author name, project ID, timestamps, feedback content, voting data, etc.

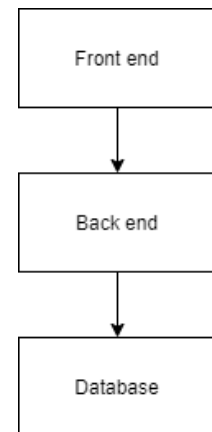


Figure 5: 3-tier layered architecture.

Below is the table recognising advantages and disadvantages of the layered architecture in the context of our project.

Advantages	Disadvantages
<b>Clear division of work:</b> The work and responsibilities are clearly divided into layers. These layers should have well-defined interfaces that are used by the adjacent layer. This creates a valuable isolation and leads to clear separation of responsibilities and division of work for developers.	<b>Lower performance:</b> The performance of a layered architecture may not be strong compared to other architectures, because some functionalities are divided into multiple layers. However, our system is not a very complex one, therefore, the performance will not be heavily affected by this property.
<b>Easy modifications of layers:</b> You can easily replace the old layer implementation with a new one. As long as the interface provided before and after is the same, it can be replaced. One major goal of this project is to improve the existing system by upgrading functionalities and replacing components. Layered architecture will make it easy for us to do so.	<b>Poor scalability:</b> Since layered architecture is monolithic by design, it can be hard to make it scalable [8]. With increasing traffic, we would need to find a way how to handle increasing number of (even concurrent) conversions. But since we are not expecting high load, this is not of concern for us right now.
<b>Low cost:</b> The overall cost of the layered architecture is low [8]. Our project is a student project which means we do not have a large budget.	

**Less dependencies:** Layered architecture can greatly reduce the dependency between system levels so that when one level changes a lot, the other level doesn't have to change much. Our project has three layers that have clear dependencies between each other that are used for different functionalities.

### 3.2 Quality requirements of the system

It is critical for us to create a system that is *changeable*, *analysable*, *acceptable* and *stable* [9].

The improved system's functionalities will be implemented in a way that they can be easily modified such as for making improvements, making corrections, and adapting to changes in the operating environment in order to meet the changing needs of the client. This will be achieved by adopting a modular code implementation architecture, such as using separate functions for each task alongside including thorough comments within them. Consequently, this approach will be greatly contributing to the system's *changeability*.

Moreover, the *back end logging* feature will contribute to the system's *analysability*, which will help the system admins/developers diagnose the causes of software malfunctions or identify areas where alterations are required to achieve the expected workflow and maintain the system's *availability*. Furthermore, *test suites* with a coverage of at least 80% will be implemented in order to achieve *stability* or rather making sure that the software can carry out tasks with adequate standards and performance, avoiding unexpected outcomes by being robustly fail-safe alongside strengthening the system's *changeability* so that later changes do not compromise any functionalities.

Finally, an *acceptable* system will be achieved by highly detailed documentation that will be produced and updated throughout the project course, which will positively affect the ability of the software to be learned, understood, used, and appreciated by ImagineThis' users.

### 3.3 Functional model

Figure 6 represents the functional model of our system. Designers and end users are the two main users of our system.

Designers need to use the web browser to get onto the ImagineThis V2 system. Then they can use the system to convert the Figma project to the prototype code using the conversion interface through the code conversion system component. They can also send and view the feedback from other end users using the feedback system component through the feedback management interface.

For the end users, our ideal use case it that they can run the code converted by the system directly on their iOS or Android devices. They can provide their feedback through the website the same way as the designers.

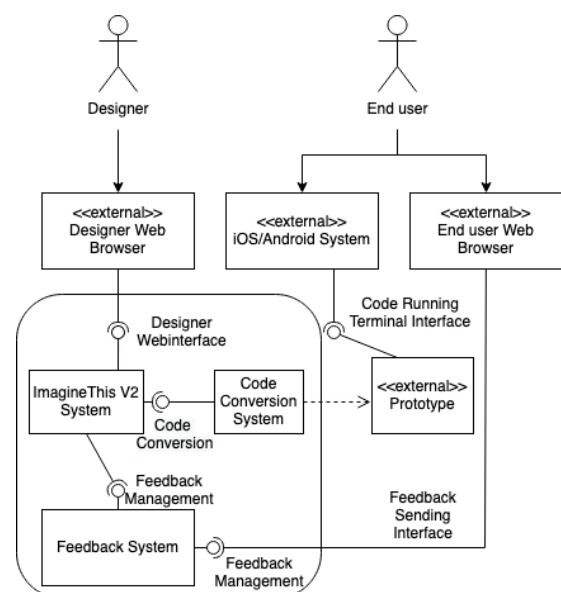


Figure 6: Functional model of ImagineThis V2.

### 3.4 Implementation of the System Architecture

As our project is based on the existing ImagineThis system, we need to specify the components that already exist and the ones we are going to develop or modify. The original system uses a 2-tier layered architecture that includes only the front-end layer and back-end layer. The front-end layer is the website system that can be accessed using the web interface and already is using back end's interface for converting the Figma project into code. This interface can be directly used by us.

We will add the feedback system as a new component into the front-end layer which can be accessed using the feedback management interface. There is no database layer in the current system, so we will need to implement this in order to store feedback data. Then, the feedback system will use the back-end layer to retrieve the feedback stored in the database.

Also, the back-end layer is responsible for converting the project which is already implemented in the current system. There is a code conversion system component that is used for code conversion. Based on such component, we will modify it to improve the functionalities and performance to achieve some of our goals.

### 3.5 Deployment

Figure 7 shows the deployment model of this project. The system will be deployed on a Linode server with Linux as the underlying operating system. Furthermore, the project will be containerised in Docker so that it can be run on any server whose CPU supports virtualisation technology. Containerisation is OS-level virtualisation, which cost less than the traditional virtual machine, while retaining advantages like easy-settable environments and cross-platform capability [10].

For the deployment of our system, both us and the client believe that this project can serve as an efficient tool to improve the speed of the development not only for the NHS but for other users and companies as well. Therefore, rather than letting each organisation to set up their own instance, the system is deployed globally on website <http://88.80.186.99/>, so that anyone can use this service if wanted; this includes users at the NHS or students at UCL hackathons.

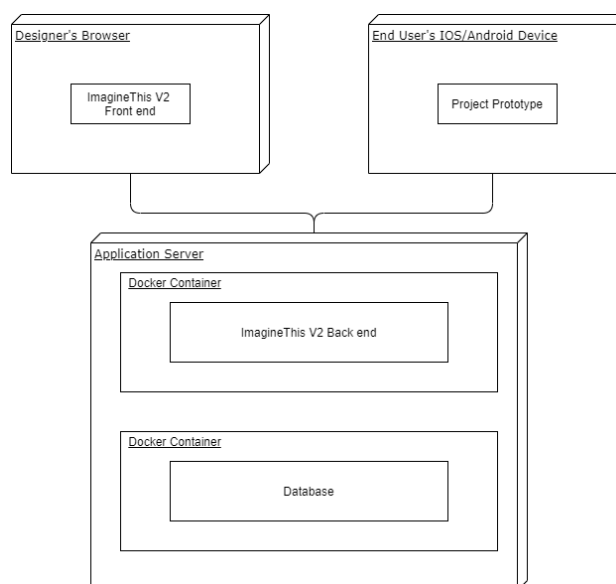


Figure 7: Deployment model of ImagineThis V2.

## 4 Development and evaluation plan

### 4.1 Development plan

As Term 2 is 12 weeks long, we are dividing it into 6 sprints, each lasting 2 weeks. Gantt chart in Figure 8 shows our timeline of how we are going to structure work. In nutshell, up until Sprint 3, we focus on developing the main new feature, the feedback system, and afterwards, from Sprint 4 onwards, we work on other independent features that do not have any prerequisites.

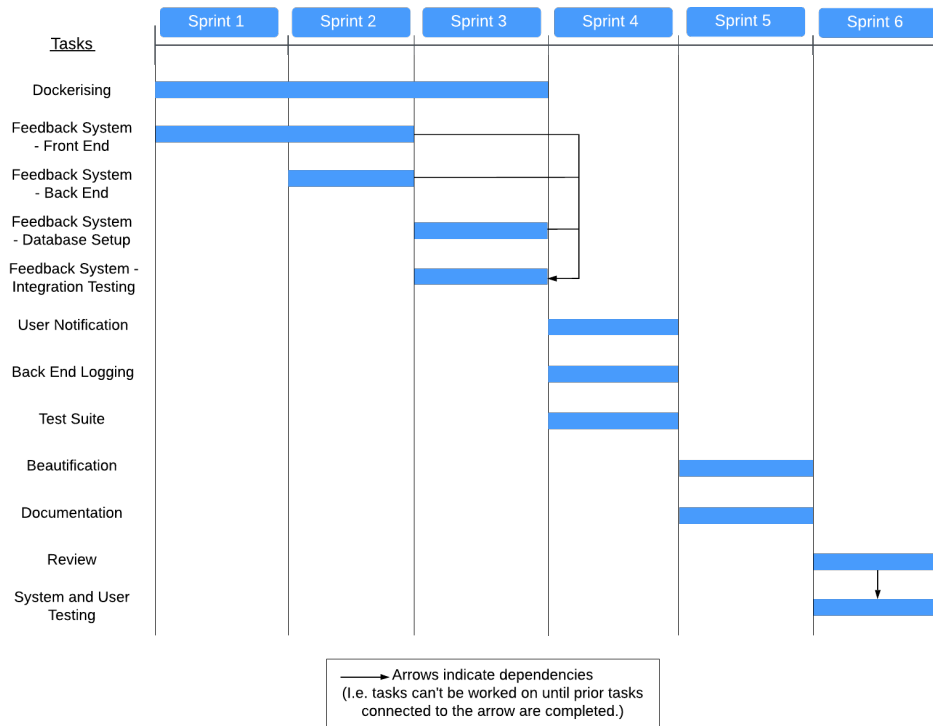


Figure 8: Gantt chart for development plan in Term 2. Each sprint represents 2 weeks.

### 4.2 Evaluation plan

We are going to evaluate our project in several ways. Some features cannot be directly measured, so instead, we will qualitatively describe the improvement. In other cases, we aim to provide quantitative data showing the improvement. In particular, we want to conduct user experience (UX) testing on few test subjects, and we want to measure statement coverage of the test suite. For all parts we have written a list of questions. Their answers will describe how successful our project was.

- **Feedback system:** Can end users write feedback on prototypes? Can designers view it? Can users upvote or downvote other feedback? How intuitive and easy are those features to use?
- **Test suite:** What is the statement coverage of back end's test suite? How has it improved the current state?
- **Dockerising:** Is the whole system containerised with Docker? Is it deployed on Linode infrastructure?
- **Documentation:** Has documentation been united into one place? Does documentation for deployment and development exist? Have there been any other improvements to the current documentation?
- **User Notification:** Can users see whether conversion was successful or failed?
- **Back End Logging:** Does the back end log its operations in failure?

- **Beautification:** Does generated code follow standard styling conventions? How does the code differ to the one being produced currently?

### 4.3 Future work

There are few features we have discussed in detail but chose not to implement primarily due to lack of time and capacity. Nevertheless, these ideas can be further investigated later, even by other student groups. These features include:

- **Interactive graphs:** We discussed with the client the need for sketching graphs, charts and data easily and converting those visualisations into code. This feature is already implemented to a certain extent. Even ImagineThis documentation [11] describes this. Currently, ImagineThis looks for a Figma component that includes the word "chart" and then generates a constant pre-programmed React Native component using a popular library React Native Chart Kit [12]. But it does not try to replicate the data or even the chart type that is used in the Figma design. This could be an enhancement. Additionally, ImagineThis could improve its support for charts even by utilising existing Figma plugins, for example Chart [13].
- **Multithreading:** In their future work section, previous group mentioned improving the time of conversion from the current approximate 15 seconds per view. They suggested using multithreading during conversion.
- **Feedback from within prototypes:** In our planned implementation, the user interface for the feedback system will be the ImagineThis website. This, unfortunately, requires the end user to remember the project ID and navigate to ImagineThis website. We envisioned another approach, where a separate React Native component would allow the user to leave feedback directly from the prototype itself. In particular, we discussed an overlay button located on top of the main view within the prototype, which the user could utilise for providing feedback. This would greatly ameliorate the system's efficiency in gathering feedback.
- **Other types of feedback:** Currently we're allowing end users to provide only written feedback, but there are other forms that can be implemented in future work to give designers a more comprehensive critique of their prototype. These examples include screenshots and capture logs of actions that users perform.
- **Security:** We discussed the use of the HTTPS protocol instead of the currently implemented HTTP. Encryption would positively impact the system's security.
- **Authorisation:** Currently anyone knowing the Figma project ID can view and add feedback, which can be a threat to the system's integrity and authenticity. Implementation of a log-in and authorisation system would prevent malicious users from gaining access to projects in ImagineThis. Then, only authorised users could view and add data.
- **Hosting prototypes on ImagineThis:** We envisioned a system where users would need just their mobile devices to test prototypes. ImagineThis would host the prototype application and users would only need to scan a QR code through Expo [14], without the need of downloading files and installing dependencies. This would significantly improve system's usability.
- **Kubernetes:** We discussed taking full advantage of ImagineThis's dockerised environment that we are going to implement and use Kubernetes as orchestration system for containers. This would be a great setup for scalability, as the system would dynamically adapt according to user load. Nevertheless, at this point, scalability is not an issue for ImagineThis.

## 5 Bibliography

- [1] [Online]. Available: <https://www.figma.com/>.
- [2] [Online]. Available: <https://www.linode.com/>.
- [3] [Online]. Available: <https://gdpr.eu/>.
- [4] [Online]. Available: <https://www.hl7.org/fhir/overview.html>.
- [5] [Online]. Available: <https://www.openehr.org/>.
- [6] [Online]. Available: <https://pitest.org/>.
- [7] M. Fowler. [Online]. Available: <https://martinfowler.com/bliki/TestCoverage.html>.
- [8] M. Richards and N. Ford, Fundamentals of Software Architecture, O'Reilly Media, 2020.
- [9] "SOFTWARE TESTING Fundamentals. 2021. Software Quality Dimensions," [Online]. Available: <https://softwaretestingfundamentals.com/software-quality-dimensions>.
- [10] [Online]. Available: <https://www.docker.com/resources/what-container>.
- [11] [Online]. Available: <https://imaginethisuc.github.io/guidelines/components/chart.html>.
- [12] [Online]. Available: <https://www.npmjs.com/package/react-native-chart-kit>.
- [13] [Online]. Available: <https://www.figma.com/community/plugin/734590934750866002/Chart>.
- [14] [Online]. Available: <https://expo.io/>.