Software Architectural Design Approval:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Issued by:  DEV |  | Sridhar Manickavel |  |  |  |  |  |
|  | Name printed |  | Signature |  | Date |  |
| Revision No.: | 01 | | | | | | |
| Approved by  PL/PM: |  | Diksha Babhoota |  |  |  |  |  |
|  | Name printed |  | Signature |  | Date |  |
| Approved by  RA/QA: |  | Sreejith Viswam |  |  |  |  |  |
|  | Name printed |  | Signature |  | Date |  |
| Approved by  Product Owner: |  | Matthew Carter |  |  |  |  |  |
|  | Name printed |  | Signature |  | Date |  |
| Approved by R&D Engineering: |  | Ezhil Ilanko |  |  |  |  |  |
|  | Name printed |  | Signature |  | Date |  |

SGTC-NPD-007, Knee Balancer

Version Number: 01

D007010025

**Table of Contents**

[1 Introduction 6](#_Toc110247312)

[1.1 Purpose 6](#_Toc110247313)

[1.2 Scope 6](#_Toc110247314)

[1.3 Business Context 6](#_Toc110247315)

[1.4 Definitions, Acronyms, and Abbreviations 6](#_Toc110247316)

[1.5 References 7](#_Toc110247317)

[1.6 System Context 7](#_Toc110247318)

[2 System Decomposition 9](#_Toc110247319)

[2.1 Hardware Decomposition 9](#_Toc110247320)

[2.2 Software Decomposition 9](#_Toc110247321)

[2.2.1 Knee Balancer App-iOS 12](#_Toc110247322)

[2.2.2 Knee Balancer Backend - Cloud 13](#_Toc110247323)

[2.3 Support for Manual Operations of the System 14](#_Toc110247324)

[2.3.1 Human-equipment Interface 14](#_Toc110247325)

[2.3.2 On-line Help Menus 14](#_Toc110247326)

[2.3.3 Speech Recognition 14](#_Toc110247327)

[2.3.4 Voice Control 14](#_Toc110247328)

[3 Risk Control 14](#_Toc110247329)

[4 Deployment View 16](#_Toc110247330)

[5 Runtime View 16](#_Toc110247331)

[6 Architectural Key Aspects 17](#_Toc110247332)

[6.1 Safety 17](#_Toc110247333)

[6.2 Accuracy 17](#_Toc110247334)

[6.3 Extensibility 17](#_Toc110247335)

[6.4 Configurability 17](#_Toc110247336)

[6.5 Maintainability 17](#_Toc110247337)

[6.6 Testability 17](#_Toc110247338)

[6.7 Security 17](#_Toc110247339)

[6.8 Performance 18](#_Toc110247340)

[6.9 Scalability 18](#_Toc110247341)

[6.10 Reliability 18](#_Toc110247342)

[6.11 Workflow Control 18](#_Toc110247343)

[6.12 Error Handling and Recovery 19](#_Toc110247344)

[6.13 Logging and Tracing 20](#_Toc110247345)

[6.14 Parallelization and Threading 20](#_Toc110247346)

[6.15 Internationalization 20](#_Toc110247347)

[6.16 Localization 20](#_Toc110247348)

[6.17 Communication between Distributed Components 20](#_Toc110247349)

[6.18 Migration 20](#_Toc110247350)

[7 Design Decisions 20](#_Toc110247351)

[7.1 Design Decision 1 20](#_Toc110247352)

[7.2 Design Decision 2 20](#_Toc110247353)

[7.3 Design Decision 3 21](#_Toc110247354)

[8 List of Software of Unknown Provenance (SOUP) 21](#_Toc110247355)

[9 Development Environment 22](#_Toc110247356)

[9.1 Standards 22](#_Toc110247357)

[9.2 Methods 22](#_Toc110247358)

[9.3 Object Oriented Design and Analysis with UML 22](#_Toc110247359)

[9.4 Tools 22](#_Toc110247360)

[9.5 Supporting Items 22](#_Toc110247361)

[9.5.1 Swift programming 22](#_Toc110247362)

[9.5.2 Continuous Integration/CD 22](#_Toc110247363)

[9.5.3 Code Reviews 22](#_Toc110247364)

[9.5.4 Unit Tests 22](#_Toc110247365)

[10 Document Revision History: 22](#_Toc110247366)

1. Introduction
   1. Purpose

This document describes the overall architectural design of the Knee Balancer Application system. It depicts from a high-level perspective of the system’s context. Where details go beyond the scope of this document the reader is referred to lower level architecture and design documents. The intended audience for this document is development, regulatory affairs, and quality assurance.

* 1. Scope

Operating principle The Knee Balancer application is a clinical decision support software tool which provides a list of possible TKA implant positions intra-operatively that achieve the surgeons desired knee laxity for an individual patient, using the initial plan and initial knee laxity values as inputs.

Please refer Intended Use document [IU] for more details.

The system supports surgical knee procedures, including: - Total Knee Arthroplasty (TKA)

1. The Knee Balancer app will be designed for iPhone and iPad
2. The app is distributed via Stryker App Store
3. Product security

App includes

* Case input – Gap, alignment angle, rotation angles, resection depths, limb varus/valgus angle and limb flexion angle
* Case solution
* Surgeon/User preferences
* Case details
* Uploading application logs, input and output parameters to cloud
* Mako robotic system screen recognition

Out of scope

* Digital connection to the other system (Bluetooth)
* Medical image data intake
* Download of case information and send email
* Record position page
* Mid resection surgeon preference
  1. Business Context

Refer Project Charter [PC] for Business Context

* 1. Definitions, Acronyms, and Abbreviations

Following definitions, acronyms, and abbreviations are used throughout this document:

|  |  |
| --- | --- |
| Terms | Definition |
| MPS | Mako Product Specialist |
| MAKO | Robotic-Arm Assisted Surgery |
| MA | Mechanical Alignment |
| FA | Functional Alignment |
| IA | Individualized Alignment |
| HKA | Hip Knee Ankle Alignment |
| STR | Soft Tissue Release |
| MVVM | Model-View-ViewModel |

* 1. References

|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **Doc. No.** |
| SDP | Software Development Plan | D007010027 |
| URS | [User](http://frwindprd02.leibinger.strykercorp.com/Windchill/servlet/WindchillAuthGW/wt.content.ContentHttp/viewContent/List%20of%20Applied%20Standards%20-%20Knee%204.1.docx?u8&HttpOperationItem=wt.content.ApplicationData%3A62928137&ContentHolder=wt.doc.WTDocument%3A62924772) Requirement Specification | D007010058 |
| SRS | SOFTWARE\_REQUIREMENTS\_SPECIFICATION | D007010059 |
| PC | Project Charter | D007010001 |
| IU | Intended Use | D007010015 |
| DDP | Design and Development Plan | D007010004 |
| RMP | Risk Management Plan | D007010012 |
| RTRM | Risk Table and Risk Matrix | D007010017 |
| SOUP | Software Soup description | D007010081 |

-

* 1. System Context

The pre-operative data or the initial plan along with the Surgeon preference values are entered into Knee Balancer application by manual input or the initial plan can be image captured from the Mako system by MPS. Based on the input values, Knee Balancer application generates solutions for the preferred gap values. MPS will discuss the available solutions with Surgeon before entering the solution position into the Mako system. Also, surgeon can assist MPS to change the input values and generate solutions again if required. Knee Balancer application will not confirm/verify the selected solution is used in the Mako surgery or not, it is out of scope for Knee Balancer application.

Knee Balancer application will not store any PHI data within the application or in the cloud.

As per the safety class evaluation of the modules in the application, Knee Balancer application is classified as “Class B”.



The main users of the system are

|  |  |  |
| --- | --- | --- |
| ID | Users | Description |
| USR1 | MPS | MPS is the user of the Knee balancer application on behalf of surgeon.  Creates initial planning and inputs the planning values to Knee Balancer application to generate gap solution. Based on the discussion with surgeon, MPS updates the values in the Mako system. |
| USR2 | Surgeon | Responsible for the reviewing and selection of final solution generated by Knee Balancer application |
| USR3 | Post Market support team | Team supports the post market complaint investigation. They are allowed to read the logs and case input/output values |

The other system which is used for the Knee Balancer application

|  |  |  |
| --- | --- | --- |
| ID | Neighbouring system | Description |
| NS1 | Mako system | Robotic system used for the Knee surgery. MPS user read the planned values for generating the solutions. After discussing with surgeon on the generated solution, MPS update the value back to Mako system. |
| NS2 | Stryker App Store | Stryker App store is used for publishing the Knee Balancer application to the assigned users. To install the Knee Balancer, user must enrol to Stryker App Store with valid Stryker User ID. |
| NS3 | Azure Cloud | Azure cloud is used for archiving the logs and case input/output to the Azure blob storage once the exited from the case. |

1. System Decomposition
   1. Hardware Decomposition

Not applicable

* 1. Software Decomposition

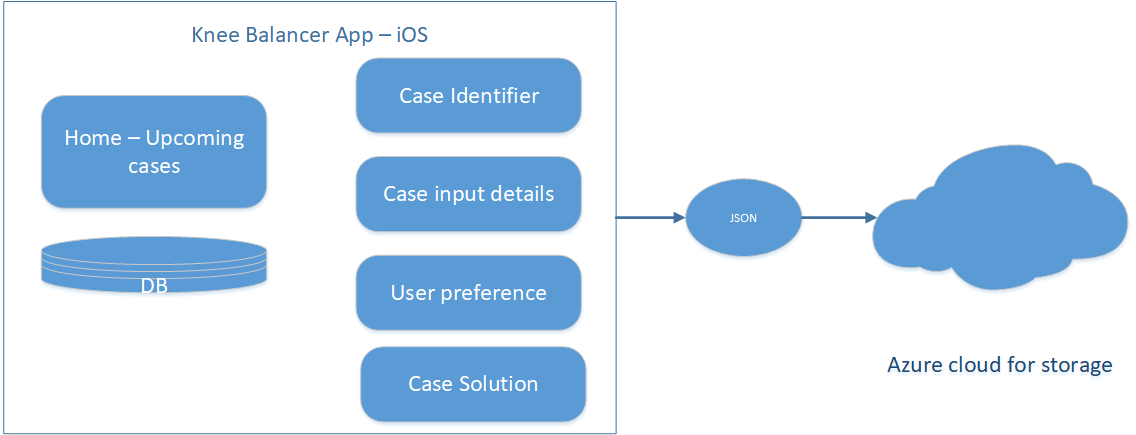
**Overall Software System**

Knee Balancer application is developed only for iPhone and iPad Stryker users. This application allows to enter the values only through manual input or through photo image where application process and reads the input values. Additionally, application archives the logs and case input/output to Azure blob storage.



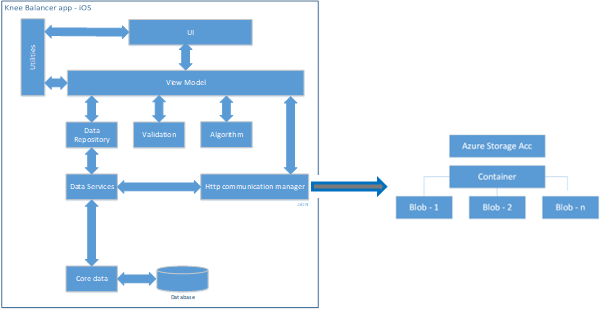
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SAD ID | Units | Description | SRS ID | Safety Class |
| SD100 | Case Planning data | Case planning is done by inputting initial input parameters | SRS-1.1.1 to SRS1.1.11 | A |
| SD101 | Knee Balancer | The Knee Balancer application is a clinical decision support software tool which provides a list of possible TKA implant positions intra-operatively that achieve the surgeons desired knee laxity for an individual patient, using the initial plan and initial knee laxity values as inputs | SRS-5.0.1 to SRS-5.0.4, SRS-6.0.0 to SRS-6.0.1, SRS-6.1.0, SRS-6.4.1 | A |
| SD102 | Core Data | The case details are stored in the local database provided by iOS platform | SRS-9.0.0, SRS-3.0.5 | A |
| SD103 | Log Files | The log files contain the errors or exception logs in JSON format | SRS-3.0.3 | A |
| SD104 | User Manual | Guides user to use the Knee Balancer application in English language | SRS-6.2.0, SRS-7.0.0, SRS-8.1.0 | A |
| SD105 | MPS | Main user of the Knee Balancer application | SRS-1.2.4 | A |
| SD106 | Azure Storage | The case details and logs are archived in the Azure | SRS-3.0.3 | A |

**Component details:**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SAD ID | Units | Description | SRS ID | Safety Class | Safety Class post external risk control measure |
| SAD100 | Home- Upcoming cases | Upcoming case will show added case details. of the required case can be found using search | SRS-1.0.2 to SRS-1.0.9, SRS-1.4.3,  SRS-1.4.4 | A | A |
| SAD101 | Case input | Case input screen allows the user to input femoral pre-operative plan data including medial distal femoral, lateral distal femoral, medial posterior femoral, lateral femoral posterior resection depths, femoral component varus/valgus and internal/ external rotation, and tibial pre-operative plan data including medial proximal and lateral proximal tibial resections and tibial component varus/valgus.  Allows to add intra operative data like medial extension gap, lateral extension gap, medial flexion gap and lateral flexion gap Or User can input through camera capture All values except mHKA parameters which needs to be entered manually. The default preference is already selected, which can be modified from the available preferences | SRS-1.1.1 to SRS-1.1.11, SRS-3.0.2 SRS-3.1.1, SRS-6.3.0 | A | A |
| SAD102 | Case solution | The automated plan will be generated considering the following inputs: pre-operative data, Surgeon Preferences (Ranges and Targets), Intra-operative data (HKA, FFD, initial gaps) set in the surgeon preference card.  Manually adjust solution parameters to view other solutions | SRS-1.2.1 to SRS-1.2.13, SRS-2.0.1,  SRS-6.4.0,  SRS-1.5.1 | B | A |
| SAD103 | Surgeon/ User preference | Allow the user to select mechanical wide, mechanical narrow, individualised wide and individualised narrow.  Also allow to customise the target gap/laxity values | SRS-1.3.1, SRS-1.3.4,  SRS-1.3.5,  SRS-1.3.6 SRS-1.3.9, SRS-3.0.1 | A | A |
| SAD104 | Case identifier | Application allows to add the following details Case Id, Hospital, Surgeon, Surgery date. | SRS-1.4.2 | A | A |
| SAD105 | Knee Balancer app – iOS database | To enter data required for the knee balancer referring Mako system and generate appropriate solutions with the help of algorithms for the surgery. Store the case details in the iOS database until the data is uploaded to cloud | SRS-1.0.8 | A | A |
| SAD106 | Knee Balancer Backend-Cloud | Upload the case data and the log files (errors, exceptions etc) from the iOS app over internet on specific interval to backend | SRS-3.0.3 to SRS-3.0.5 | A | A |
| SAD107 | Knee Balancer Authentication | Enables application to acquire tokens from the Microsoft identity platform in order to authenticate users and access secured web APIs. | SRS-6.5.0 to 6.5.2, SRS-8.0.0 | A | A |
| SAD108 | Algorithm | Algorithm will take the input parameters and surgeon preference to generate list of solutions which can be reviewed by the surgeon to choose the most clinically relevant..  The algorithm to consider the parameters based on the TKA 1.0 and TKA 2.0 selection from the user. For TKA 2.0 considers the laxity values and for TKA 1.0 considers gaps values to generate list of solutions | SRS-1.5.1 | B | A |

The block diagram of the architecture is shown



* + 1. Knee Balancer App-iOS

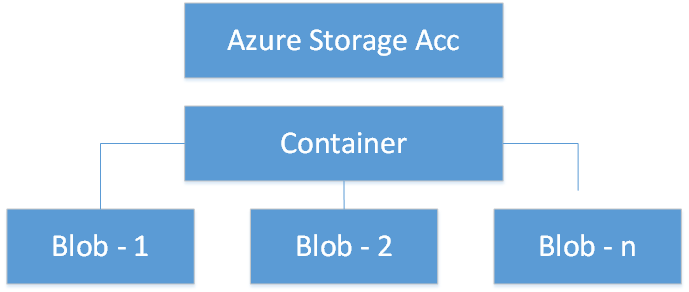
The frontend Knee Balancer application diagram is as shown



|  |  |
| --- | --- |
| Units | Description |
| App View/UI | Module is responsible for displaying the view to the end users and enables them to perform user interface events. The UI layer is created using storyboard provided by apple. And each of the UI element is associated with UI controller. |
| View Model | This module is responsible for handling all business logics, converting data in model into human readable format that can be presented in the view-by ViewController.  Updates to View data would not go directly to the Model, rather they would  be triggered by the view controller talking to the View Model which would then talk to the model. |
| HTTP communication manager | This module is responsible for interacting with web server for sending /receiving (JSON) data using the REST API and communicates back to view model, data repository |
| Data Repositories | This module is acts as a wrapper for all database services, so that your application can work with a simple abstraction that has an interface. |
| Database Services | This module is responsible for handling the data base operation implementation and sending back the result into ViewModel again. |
| CoreData | Core Data is a framework that use to manage the model layer objects in application. It provides generalized and automated solutions to common tasks associated with object life cycle and object graph management, including persistence. |
| Validator | This module is responsible for validating the input provided and sending back the validated output into the view model. |
| Utility | Used to assist in providing some reusable functionality (Colour literals, custom fonts etc..) |
| Database | Local database within the iOS for the local storage to store data temporarily when executed offline until the data is sent to cloud database |
| Algorithm | This module contains all of the logic for solving an initial position within the set limits and boundaries. Takes an InitialPosition object, preference object stores results as an array of Solution objects. |

* + 1. Knee Balancer Backend - Cloud

The Knee Balancer Backend – Cloud diagram is as shown



|  |  |
| --- | --- |
| Units | Description |
| Azure Storage Account | Azure storage account contains all Azure Storage container and objects i.e files. The storage account provides a unique namespace for your Azure Storage data that is accessible from anywhere in the world over HTTP or HTTPS. |
| Azure Blob Service | To store the data as json file in Azure cloud-azure blob storage for future retrieval and analysis |

* 1. Support for Manual Operations of the System

The application allows the user to enter input parameters of pre-operative data and the intra operative data manually through incrementing/ decrementing numbers through buttons or through capturing MAKO photo through camera feature

* + 1. Human-equipment Interface

Not applicable

* + 1. On-line Help Menus

Not applicable

* + 1. Speech Recognition

Not applicable

* + 1. Voice Control

Not applicable

1. Risk Control

The risks are captured in the Risk Management Plan [RMP] and Risk Table and Risk Matrix [RTRM]

|  |  |  |  |
| --- | --- | --- | --- |
| **Units** | **Safety Class Before Risk Controls** | **Rationale (How are software functions segregated and risk controls effective to allow classification into another safety class?)** | **Safety Class After Risk Controls** |
| Algorithm | B | Algorithm take the pre-operative values as input and surgeon preference to generate list of solutions which will be reviewed by Surgeon. Based on the discussion between Surgeon and MPS, values will be entered into Mako system and Surgeon will assess intra-operative and adjust the values as per the assessment. The Knee Balancer application is intended as a clinical decision support system that provides planning assistance forTotal Knee Arthroplasty (TKA) procedures. The application is not intended to replace the clinical judgement of the health care professional. The healthcare professional can exercise their own judgement in determining whether to accept the recommendation. Since some of the software component failure can leads to hazardous situation the initial classification will falls under class B but with control and review by the surgeon to evaluate and make changes in the final output, the final software safety class will be class A as per the guideline IEC 62304. | A |
| Case solution | B | Based on the solutions generated by the algorithm, Surgeon and MPS has option to review all the solutions. Based on the review Surgeon will assess intra-operative and adjust the values as per the assessment. The Knee Balancer application is intended as a clinical decision support system that provides planning assistance forTotal Knee Arthroplasty (TKA) procedures. The application is not intended to replace the clinical judgement of the health care professional. The healthcare professional can exercise their own judgement in determining whether to accept the recommendation. Since some of the solutions can leads to hazardous situation the initial classification will falls under class B but with full control and review by the surgeon to evaluate and make changes in the final solution selection and decision. The final software safety class will be class A as per the guideline IEC 62304. | A |

Based on the above rational, the final software classification for Knee Balancer is classified as “Class A”

Refer below flow chart from IEC 62304:2015



1. Deployment View

The app is deployed in the Stryker app store where the MPS users can download the application via MS Intune company portal. This is secured portal and helps to manage and access apps securely to the Stryker users within the organization.

To deploy the Knee balancer app to the Stryker app store, need to perform following

* Generate release build from the iOS code base
* Use Apple account to generate release build
* Use Certificates for the distribution
* Generate ipa file and publish it over Stryker app store

1. Runtime View

The app runs on the iOS platform with OS version minimum 14.5.x. The target devices to run the app are

* iPhone 13
* iPhone 13 mini
* iPAD Air 4th gen

1. Architectural Key Aspects
   1. Safety

There is no component with safety classification C. Refer section 3 for the component level classification.

* 1. Accuracy

The Knee Balancer app employs the resection algorithm which is a proprietary Stryker algorithm. There is a geometric relationship between the implant position, bone resections and knee laxity values.  It takes inputs from the Mako system and generates a list of solutions that achieve the surgeons desired knee laxity targets. Surgeon can verify the solutions and make appropriate changes (if required) based on clinical judgement and proceed. The Knee Balancer application is not intended to replace the clinical judgement of the health care professional. The healthcare professional shall exercise their own judgement in determining whether to accept the recommendation

* 1. Extensibility

The Knee Balancer application follow MVVM design pattern which allows to develop UI and business logic separately. It allows to extend additional modules in future development without impacting current design.

* 1. Configurability

User can customize their own preference as per the surgeon practice. Also allow to configure the preferences to multiple surgeons for MPS. The knee balancer app cloud service is configured with storage account and container for the Stryker Post market support team and for the Australia region.

* 1. Maintainability

The Stryker app store allows users to update software via the MS Intune company portal. There is no separate maintenance activities within the application.

* 1. Testability

As per the MVVM design pattern, components are loosely coupled so that testing can be performed at each layer.

The app uses Stryker apple account and development certificate from which it can be installed on test devices for testing

The application creates solutions and keeps it in JSON file which helps tester to compare it with the actual set of calculations offline

* 1. Security

Knee Balancer application is distributed through Stryker App Store and only Stryker user can download the application. The device is issued and maintained by the Stryker, and it polices on the device.

For uploading the logs and case details to cloud Azure AD is used for authentication.

App uses Microsoft authentication for login, Camera for Vision Kit, Core Data to store the value in encrypted mode locally. Stryker algorithm files securely stored in the app directory and path is not exposing . All Resources are stored in App directory and No one has access to modify those resources. App will be published over MS Intune company portal which is secured

* 1. Performance

The knee balancer application takes input parameters and provides list of solutions within 10 seconds

* 1. Scalability

Stryker App store supports to deploys the applications to all the users and as per the Microsoft Intune design it can extend to multiple regions and countries.

Azure service scalable to support the uploading of the logs and case details in parallel. Current estimated usage of upload will be approximately 1500 cases per user in a year.

There is no scalability requirement for Knee Balancer application since it is single user per device.

* 1. Reliability

Knee Balancer application can be reopened within few mins post the failure. Additionally, application will check for any low memory and notify the user to avoid any crash.

* 1. Workflow Control

The Knee balancer app has following workflows

1. Regular workflow



1. Quick workflow



* 1. Error Handling and Recovery

Using the Swift language for error handling and storing the logs in the files as part of the document directory

* 1. Logging and Tracing

Using the Swift language, logs are updated locally in the file as part of document directory. As part the archival, these logs will be pushed to cloud for future investigation.

* 1. Parallelization and Threading

The application allows multi-threading and parallel operations. While the regular workflow is in process and if there is past case available waiting for upload, the upload operation runs in parallel on separate thread without interfering with the regular workflow

* 1. Internationalization

The application shall support only English language and intended for SSP (Australia and New Zealand) region.

* 1. Localization

The application shall support English language only

* 1. Communication between Distributed Components

NA

* 1. Migration

Upgrading from one version to another will also upgrade the data to current version.

1. Design Decisions

* 1. Design Decision 1

The decision was to choose MVVM over MVC for the frontend architecture due to following comparison

|  |  |
| --- | --- |
| **MVVM** (Model-View-ViewModel) | **MVC** (Model-View-Controller) |
| Makes the view controller simpler by moving a lot of business logic out of it. | All business logic and UI related logics will be in ViewController class. |
| The view model better expresses the business logic for the view. | There is no ViewModel. |
| Breaks the coupling between the application logic and the UI and so make testing more accessible. | All business logic and UI related logics will be in ViewController class makes testing difficult. |
| The responsibilities of the view controller are reduced to controlling the interaction between the view layer and the model layer. | ViewController will be responsible for communicating with View and Model. |

* 1. Design Decision 2

The decision was to choose Vision Kit over Opensource OCR for the frontend architecture due to following.

One of Vision’s many powerful features is its ability to detect and recognize multilanguage text in images. You can use this functionality in your own apps to handle both real-time and offline use cases. In all cases, all of Vision’s processing happens on the user’s device to enhance performance and user privacy.

Vision’s text-recognition capabilities operate using one of these paths:

***Fast path:***

The fast path uses the framework’s character-detection capabilities to find individual characters, and then uses a small machine learning model to recognize individual characters and words. This approach is like traditional optical character recognition (OCR).

***Accurate path:***

The accurate path uses a neural network to find text in terms of strings and lines, and then performs further analysis to find individual words and sentences. This approach is much more in line with how humans reads text.

**Cost:**

Apple vision kit is free of cost compared to Google OCR (ML kit)

* 1. Design Decision 3

The decision was to choose Microsoft oAuth over other oAuth for the frontend architecture due to following.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Registration** | **Verification** | **Scopes** | **Access token** | **Time to go live** |
| **Google** | Complex, full of pitfalls | Manual | Pre-registered for verification. Given at runtime to allow for least access | String | Days to weeks |
| **Microsoft** | Complex | Automated | Pre-registered. Given at runtime to allow for least access. Also, Stryker Azure AD can be leveraged. | Identity JWT | Minutes to hours |
| **Yahoo** | Simple | None | Pre-registered. All-access is requested at once. | String | Minutes |

1. List of Software of Unknown Provenance (SOUP)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Title | Manufacturer | Version | Release Date | License Type | Maintenance procedure |
| MSAL (Microsoft Authentication libirary) | Microsoft | 1.2.2 | Jun-2022 | MIT | Not applicable |
| AZSClient (Azure Storage client library) | Microsoft | 0.2.6 | 01-Mar-2018 | MIT | Not applicable |
| MBProgressHUD | Matej Bukovinski | 1.2.0 | 13-Jan-2020 | MIT | Not applicable |

Refer [SOUP] document for more details

1. Development Environment
   1. Standards

Not Applicable.

* 1. Methods

As per SDP document [SDP] for all the development tools. Other practices are updated in the section 6.

* 1. Object Oriented Design and Analysis with UML

Will be detailed in the detail design documents.

* 1. Tools

The list of development tools is captured in SDP document [SDP].

l

* 1. Supporting Items
     1. Swift programming

Swift is a powerful and intuitive programming language for iOS. Swift eliminates entire classes of unsafe code. Variables are always initialized before use, arrays and integers are checked for overflow, memory is automatically managed, and enforcement of exclusive access to memory guards against many programming mistakes. Syntax is tuned to make it easy to define

* + 1. Continuous Integration/CD

After the PR approval the build in generated using Azure DevOps. The build is generated and published to Stryker app store for distribution.

* + 1. Code Reviews

The code reviews are done using bit bucket. Initial code review is done by the developers and raise request for PR approval. Based on the code review comments the changes are updated and the PR is approved in bit bucket

* + 1. Unit Tests

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It is performed by white box testing method. This is the first level of testing performs before integration testing which is performed mainly by the developers. XCTest framework is used for unit test.

1. Document Revision History:

|  |  |  |  |
| --- | --- | --- | --- |
| Revision  Level | Revision Date | Effective Date | Reason and Description of Revision |
| 0 | 21-Jun-22 | 21-Jun-22 | Initial draft |
| 01 | 20-Jul-22 | 20-Jul-22 | Sec 1.5 - Included SOUP  Sec 2.2 - Modified diagram with MAKO screen photo direction  Sec 6.2 - Updated accuracy section  Sec 8 - Provided SOUP reference documentation  SD101, SD104, SAD100, SAD101, SAD102, SAD103, SAD106, SAD107 - Mapped to proper SRS IDs  SAD100 - Included case search  SAD101 - Included preference selection  SAD103 – Updated SRS IDs. Removed SRS1.3.2, SRS1.3.3, SRS1.3.7, SRS1.3.8.  SAD108 - SRS-1.2.0 is removed as it is NA and included SRS1.5.1  SD102 - Mapped with SRS-3.0.5  SAD102 - Mapped with SRS-1.5.1 |