D6.3 Smartphone enhanced traveler verification 1

Document Due Date:	31.05.2020 (M9)
Document Submission Date:	29.05.2020 (M9)

Work Package 6:
Smartphone based enhanced traveller verification

Document Dissemination Level:	Public

D4FLY - **D**etecting **D**ocument frau**D** and i**D**entity on the fly
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833704

Abstract

The main objective within the work package WP6 of the D4FLY research project is to explore alternative technologies to people identification. Within this work package task 6.3 focusses on research and development of a smartphone application that can be used in coaches or similar scenarios to access pre-verified data that has been enrolled at a pre-boarding kiosk. This kind of process, using pre-enrolled and pre-verified data for traveller verification at the border using a mobile device like a smartphone, is not being used as of today and aims at making the border check process more efficient.

The smartphone application, which is being developed in this task, shall have the capability to make use of pre-enrolled data that is downloaded to the smartphone of the border guard. The smartphone camera shall be used for verifying the travellers in the coach versus the pre-enrolled data using 2D face verification technology, which matches to the pre-enrolled image in the downloaded dataset. The result of the matching is displayed on the smartphone to the border guard to enable him/her to perform risk assessment and complete the checking process as desired.

This deliverable describes the initial architecture and the implementation concept, how the smartphone application is embedded in the coach scenario and initial implementation details. Field testing of the initial version of the application in the context of the overall coach scenario is planned after the submission of this deliverable, to collect user feedback. Further improvements are anticipated based on the user feedback and further refinements. The final version of the smartphone application shall be described in the subsequent deliverable "D6.8 – Smartphone enhanced traveller identification 2", which is due in M24.

Project Information

Project Name	Detecting Document frauD and iDentity on the fly
Project Acronym	D4FLY
Project Coordinator	Veridos GmbH
Project Funded by	European Commission
Under the Programme	Horizon 2020 Secure Societies
Call	H2020-SU-SEC-2018
Topic	SU-BES02-2018-2019-2020 Technologies to enhance border and external security
Funding Instrument	Research and Innovation Action
Grant Agreement No.	833704

Document Information

Document reference	D6.3
Document Title	**Smartphone enhanced traveller verification 1**
Work Package reference	WP06 Alternative technologies to identifying people
Delivery due date	31.05.2020 (M09)
Actual submission date	29.05.2020 (M09)
Dissemination Level	Public
Lead Partner	Veridos (VD)
Author(s)	Ananya Verma, Armin Reuter (VD)
Reviewer(s)	James Ferryman (UoR) Bartłomiej Jankiewicz (WAT)

Document Version History

Version	Date created	Beneficiary	Comments
0.1	19.12.2019	Veridos (VD)	Initial draft version
0.2	17.04.2020	Veridos (VD)	First draft version
0.5	11.05.2020	Veridos (VD)	Internal review done, ready for 2nd review
0.6	19.05.2020	UREAD	UREAD internal review
0.7	20.05.2020	Veridos (VD)	Final draft version for 3rd review
1.0	25.05.2020	Veridos (VD)	Edits based on 3rd review, final version
1.1	25.06.2021	Veridos (VD	Small edits based on review comments

List of Acronyms and Abbreviations

ACRONYM	EXPLANATION
BLE	Bluetooth Low Energy
EC	European Commission
EU	European Union
D4FLY	Detecting Document frauD and iDentity on the fly
GA	Grant Agreement
GDPR	General Data Protection Regulation
ICAO	International Civil Aviation Organization
JSON	JavaScript Object Notation
MRZ	Machine Readable Zone
OCR	Optical Character Recognition
OS	Operating System
PROTECT	Pervasive and UseR Focused BiomeTrics BordEr ProjeCT
SAB	Security Advisory Board
SDK	Software Development Kit
TLS	Transport Layer Security
TRL	Technology Readiness Level
VD	Veridos GmbH
WP	Work Package

Table of Contents

1 INTRODUCTION

1.1 Background

The D4FLY project will enhance current capabilities and capacities of border guards by providing methods and tools for enhanced document and identity verification. Further on D4FLY aims at making the required border checks more efficient and effective for border guards. From the four scenarios, which are described in the GA for D4FLY, the smartphone application which is described in this deliverable targets a scenario, where travellers are approaching a European land border by coach. In order to allow the border guards at these types of borders to focus more on high risk travellers or groups of travellers, the described smartphone application aims at making the checking process for low risk travel groups travelling in organised tours in coaches more efficient and more secure.

1.2 Aim of this document

According to the "Description of the action" [1] the aim of this report is to describe "the design, development and evaluation of a smartphone application acting as a traveller verification device.". The report will present the overall architecture, the components involved, processes involved and which technical requirements are to be met in order to achieve the goal. Also, it contains a description of the applications software architecture and interfaces to other system components and the user.

1.3 Input/output to this document

1.3.1 Requirements

A first set of high level requirements related to the overall coach scenario, where the smartphone application is to be used, has been collected and defined based on an analysis of the D4FLY GA, initial discussions with partners and end users in the consortium, as well as an analysis of requirements from other projects in the same area, including PROTECT.

The collection and consolidation of specific and detailed requirements for the D4FLY system, based on user needs and further inputs from stakeholders is ongoing at the time of writing of this deliverable. They will be related to in the second deliverable related to this topic on smartphone enhanced traveler verification towards the end of the project.

These requirements are :

- The smartphone application shall be executable on a Smartphone with an Android or iOS support, which has Wi-Fi or mobile data capabilities along with a touchscreen
- The smartphone shall have a camera, which is capable of capturing a 2D image.
- The smartphone application shall be accessible only for authorised personnel (border guards).
- The smartphone application shall be installable using standard Android mechanisms.

- The smartphone application shall store all personal data only in an encrypted way and compliant to the GDPR.
- The smartphone application shall have a User Interface (GUI), which is easy to use and can be interacted with using the touch screen of the smartphone.
- The smartphone application shall be able to scan the data page of an EU passport and recognize the MRZ components using its built-in camera.
- The smartphone application shall be able to do a face verification in under 5 seconds.
- The smartphone application should perform the MRZ recognition in less than 2 seconds.
- The smartphone application shall delete all traveller related data sets after the verification process is ended (by the border guard) or the application is closed.

The concept of the smartphone application has been designed considering data protection, data privacy aspects, IT security as well as ethical aspects. In the course of the project, all these aspects are continuously monitored and will be reviewed in a privacy, data protection social and ethical impact assessment, which will be reported in separate public deliverables from work package 3.

1.3.2 Outputs

This document serves as specification and description of processes, components and interfaces related to the smartphone application that is to be used in the prototype D4FLY system configuration for the coach scenario. As such it serves as input to the design and the implementation of the smartphone app, the implementation of the system prototype for the coach scenario and as input for the planning and execution of the related field tests and demonstrations.

2 SCENARIO AND USER PERSPECTIVE

This section contains a high level description of the coach scenario in which the smartphone application is used. It describes how the process in general is envisaged and how travellers and border guards are expected to interact with the system in this scenario.

2.1 Scenario description

The scenario consists of three main parts: the pre-boarding enrolment, the backend system and the smartphone application for the verification. The enrolment will take place before the traveller boards the coach and the verification will take place inside the coach by a border guard using a smartphone application. The data enrolled during this process is stored in the backend. The graphic below describes the whole scenario in brief.

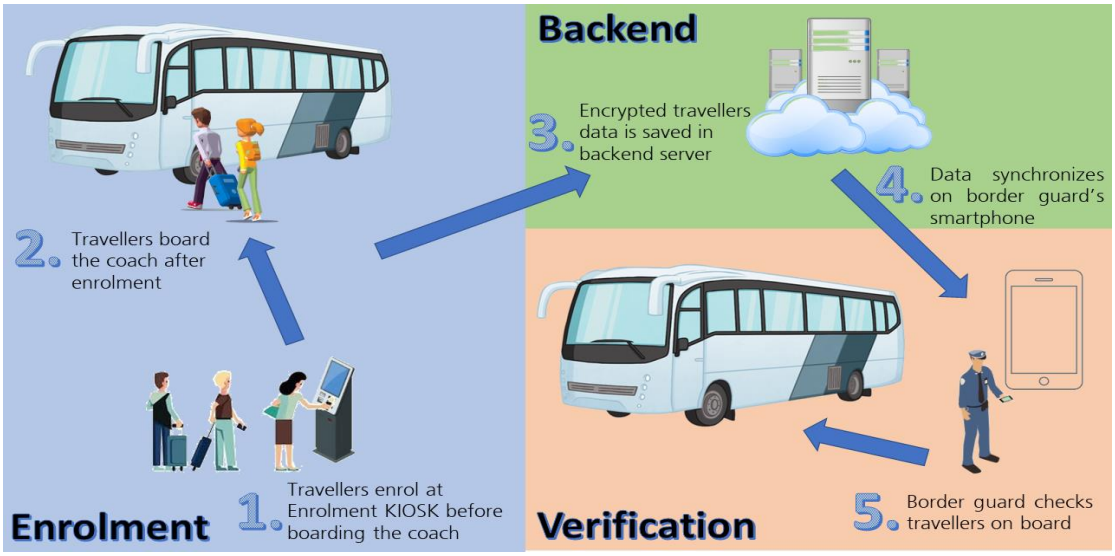

FIGURE 1: OVERVIEW OF SCENARIO

The main steps of the process are shown in Figure 2:

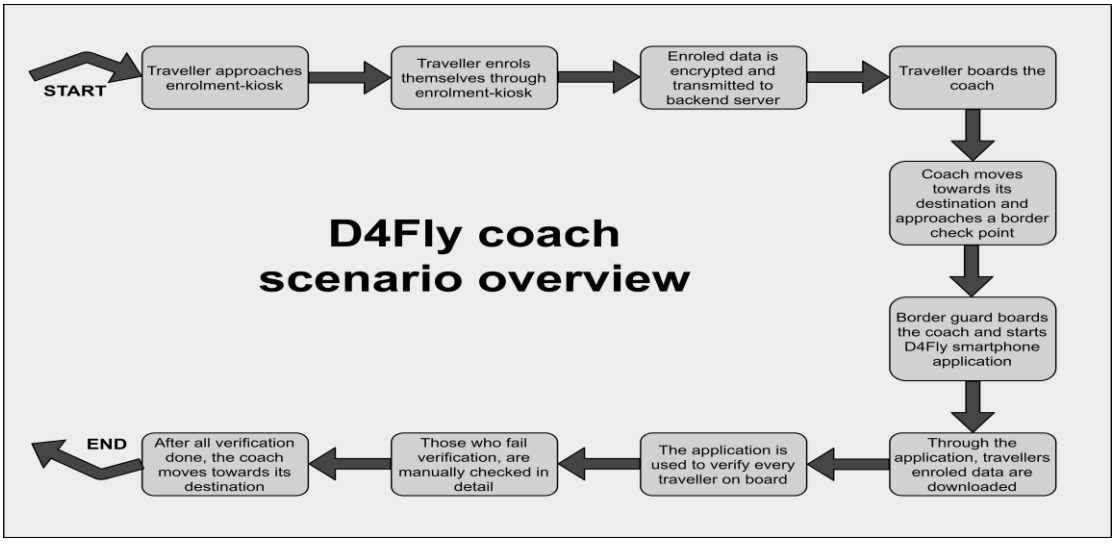

FIGURE 2: OVERVIEW OF MAIN STEPS IN THE SCENARIO

As it can be seen from Figure 1 and Figure 2, the smartphone is the main tool the border guard uses to verify the travellers that have enrolled to the system and boarded the coach. The application also interacts constantly with the border guard to assist in the verification process. It also specifies necessary steps that have to be taken in case of abnormalities. All this can happen in a mobile setting without a need of a static verification area using pre-enrolled information and data. This is one of the key advantages of the new design.

2.2 User perspective

As it can be seen from Figure 1, the entire scenario is divided into three main parts, namely: enrolment (pre-boarding enrolment), backend and verification (smartphone application). Below each part of the scenario is described from a user point of view.

2.2.1 Pre-boarding enrolment

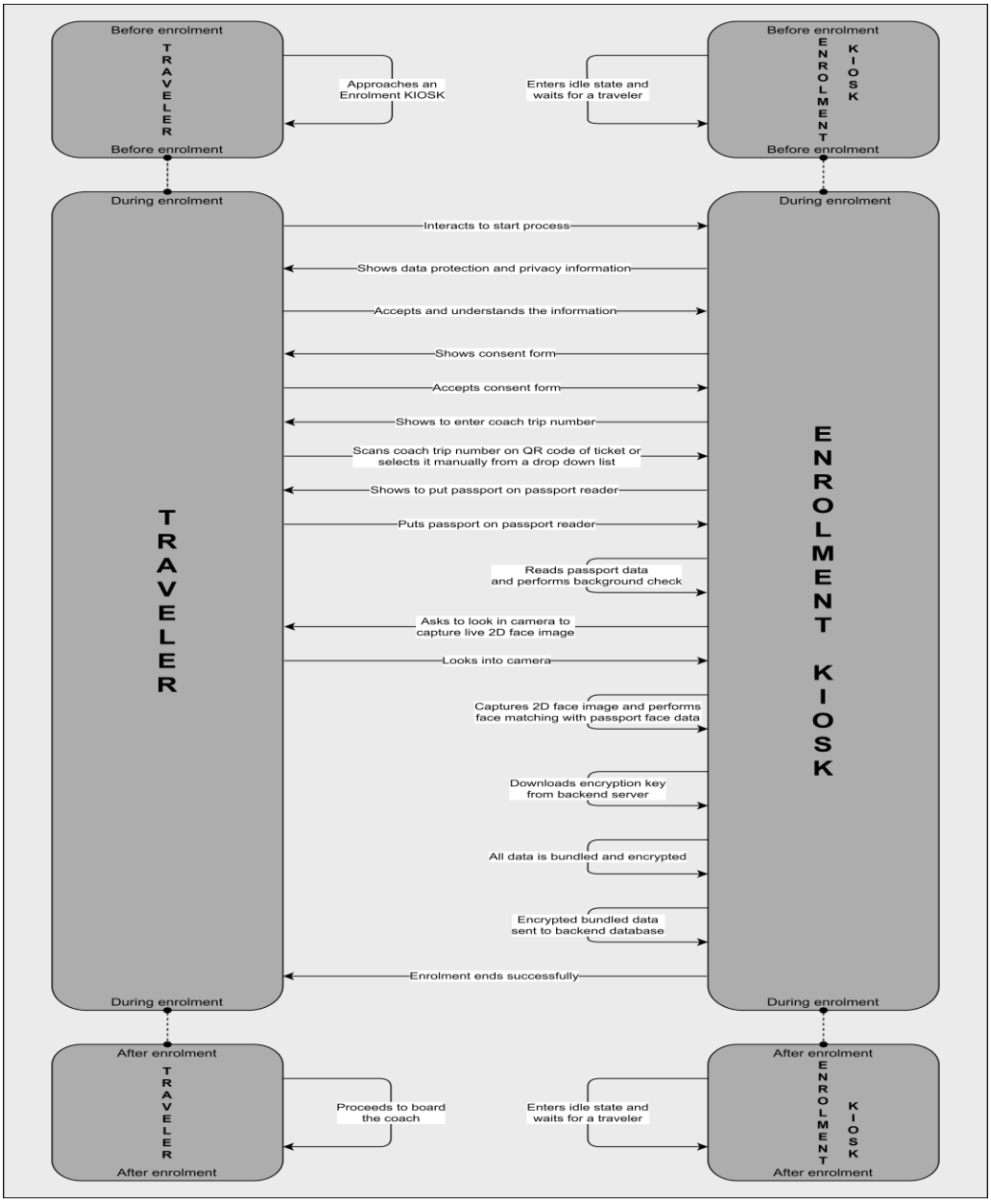

FIGURE 3: PRE-BOARDING ENROLMENT STEPS (ONLY SUCCESS SCENARIO)

The graphic above (Figure 3) describes the pre-boarding-enrolment which shows the interaction between enrolment-kiosk and traveller. Information displayed on the kiosk screen leads the traveller through these steps.

The enrolment is performed by an enrolment-kiosk. The salient features of the kiosk are:

- The enrolment kiosk will have a passport reader built-in. The traveller, before boarding the coach, puts their passport on the reader to enrol themselves. The kiosk then reads the passport, performs additional steps like pre-checks and then stores this data in the backend database.
- All data is being encrypted by the kiosk before it is stored in the backend database.
- The kiosk will also be equipped with two cameras: The first camera will be responsible for capturing the QR code from the traveller's ticket to assign the traveller to a particular coach and journey. The second camera will be responsible to take a high quality live face image to be matched with the traveller's face data stored on the passport chip. This ensures the authenticity of the traveller.

2.2.2 Backend

The backend is responsible for two main tasks: storage of traveller's enrolled data along with storing border guard login details and for helping and managing encryption.

- **Databases**
  There will be two databases set up: The first database will contain login credentials of all border guards who are registered to use the smartphone application. The second database will act as the storage of data between enrolment and verification processes. The second database will receive data during the enrolment process to be stored from enrolment kiosk. From this database the data is downloaded to the smartphone application during the verification process.

- **Encryption**
  A module called "Crypto provider" will be used to assist in encryption. It will perform two important tasks: (1) creation of encryption key (asymmetric encryption) for every coach journey to assist in encryption and decryption; (2) to store the key securely in an internal database. For every journey, the module generates a key and stores it. This key is downloaded by an Enrolment-KIOSK to encrypt the traveller's data before uploading the data to the database in the backend. The key is then also downloaded securely by the border guard's smartphone application to be used for decryption of the data sets after download.

2.2.3 Smartphone application (Verification)

The graphic below (Figure 4) describes the in-coach-verification scenario. The steps are undertaken by the border guard using a smartphone application.

The verification is performed by a smartphone application that has been installed and setup on the border guards' smartphone. The features of the application are:

- The application is protected against unauthorised use by requiring a login with correct credentials.
- The application can only display useful data, when the correct decryption key is available on the smartphone. This key is only downloaded to the smartphone

application after successful authentication, that the request has been made from a valid border guard account to the backend server.

- The application uses the built in camera to identify and verify travellers in the coach.
- The application keeps a count of travellers which have been verified, travellers which have registered but not boarded the coach, and travellers that failed verification.
- The application also has the ability to scan the MRZ of a traveller's passport and access and display the pre-enrolled dataset based on the recognised MRZ.
- The application continuously offers guidance to the border guard to assist in the entire verification process.
- The decryption key and traveller's data are only stored temporarily on the smartphone after the download from the backend and are deleted from smartphone memory once the process is completed.

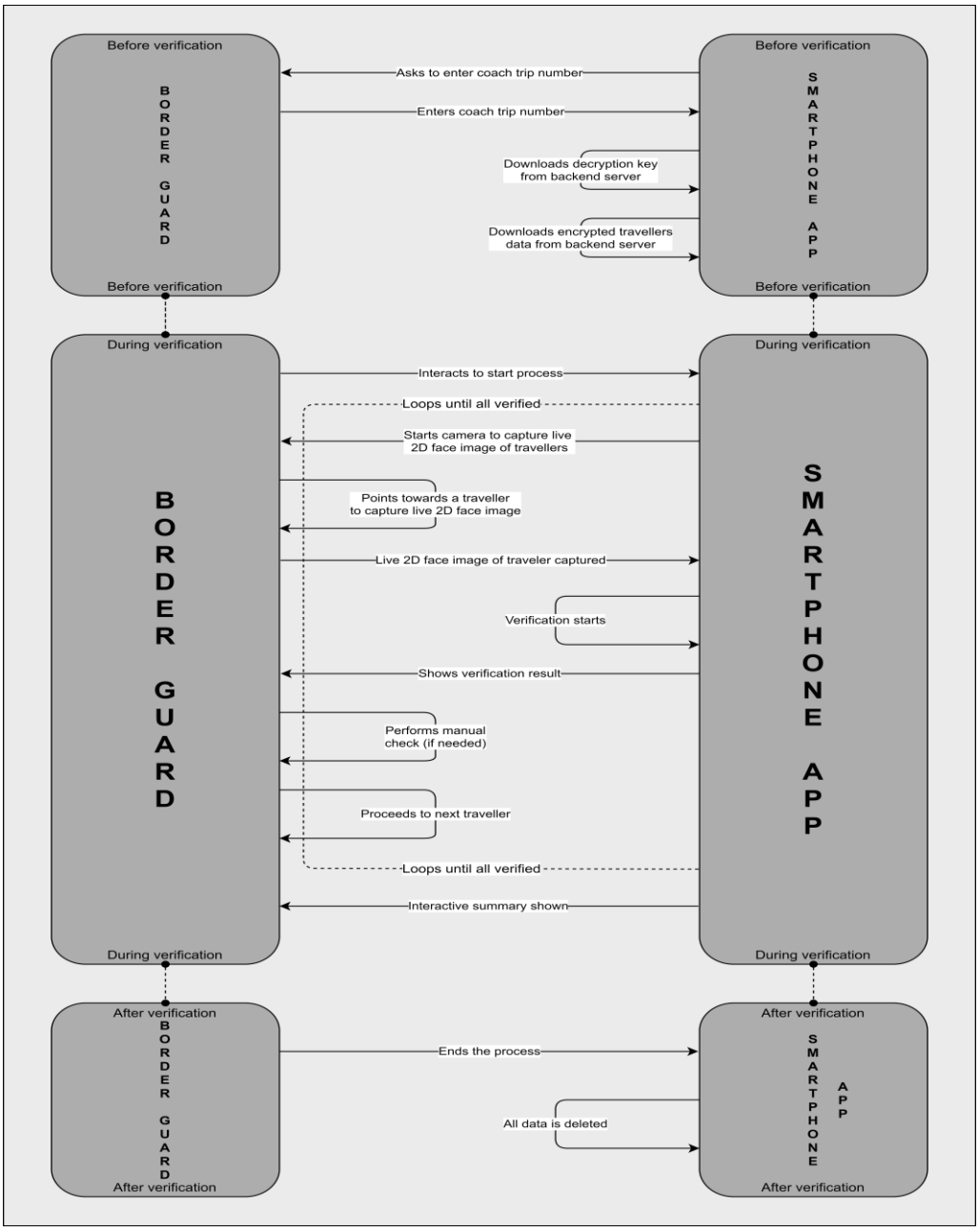

FIGURE 4: IN-COACH VERIFICATION STEPS (ONLY SUCCESS SCENARIO)

Before the app can be used, the border guard has to login with their credential. This credential is securely given to the border guard separately and is needed to work with the application. After the border guard logs in successfully, the actual verification process can be performed. As can be seen from Figure 4, following are the steps that take place:

Step 1-> The coach approaches the border.

Step 2-> The border guard approaches the coach, starts their mobile device followed by starting the D4FLY application and logging into the application

Step 3-> Then the border guard enters the coach trip number in the smartphone application. The coach trip number is the unique number that tells on which coach and which journey the traveller has booked themselves for. It can also be selected from a drop down list available in the app.

Step 4-> The application downloads the decryption key from backend server.

Step 5-> The application then downloads the encrypted travellers' data stored in backend database. Only the data of travellers in this particular coach are downloaded (referenced with help of the coach trip number).

Step 6-> The data is temporarily stored on smartphone in encrypted form and only decrypted when a data is to be used (in further steps). Data in decrypted format is only temporarily stored in run-time memory when its needed and never anywhere else on the smartphone.

Step 7-> The application opens the camera and is ready to start verification.

Step 8-> The border guard enters the coach and walks through the coach pointing the camera to the passengers one by one.

Step 9-> Images are captured using the smartphone camera and processed by the matching software. No image is permanently stored on the smartphone. As soon as a face is detected in the stream of images, it is matched against all face templates present in the data set that has been downloaded to the smartphone (matching one-to-many).

Step 10-> As soon as a match is confirmed, the traveller is shown as verified along with their passport data.

Step 11-> In case no match is found, verification failed is shown and the reading of the MRZ of the traveller's passport can be used instead, as described in the next section.

Step 12-> After all the travellers have been checked successfully, the border guard terminates the application (all data is deleted from smartphone), leaves the coach and the coach continues its journey.

In cases where face verification fails for a traveller, the following steps can be performed which is also described in the graphic below (Figure 5):

Step 1-> The border guard selects manual MRZ reading option. This opens the camera and enables it to capture MRZ.

Step 2-> The border guard then picks up the traveller's passport and points the camera towards the MRZ region of the passport.

Step 3-> The software application automatically detects the MRZ and reads it.

Step 4-> The MRZ data is then checked in the datasets to find the data of this particular traveller.

Step 5-> The data is fetched and displayed on screen which can be used by the border guard to assist in the checking procedure, e.g. to display the pre-enrolled image for an additional visual comparison with the traveller.

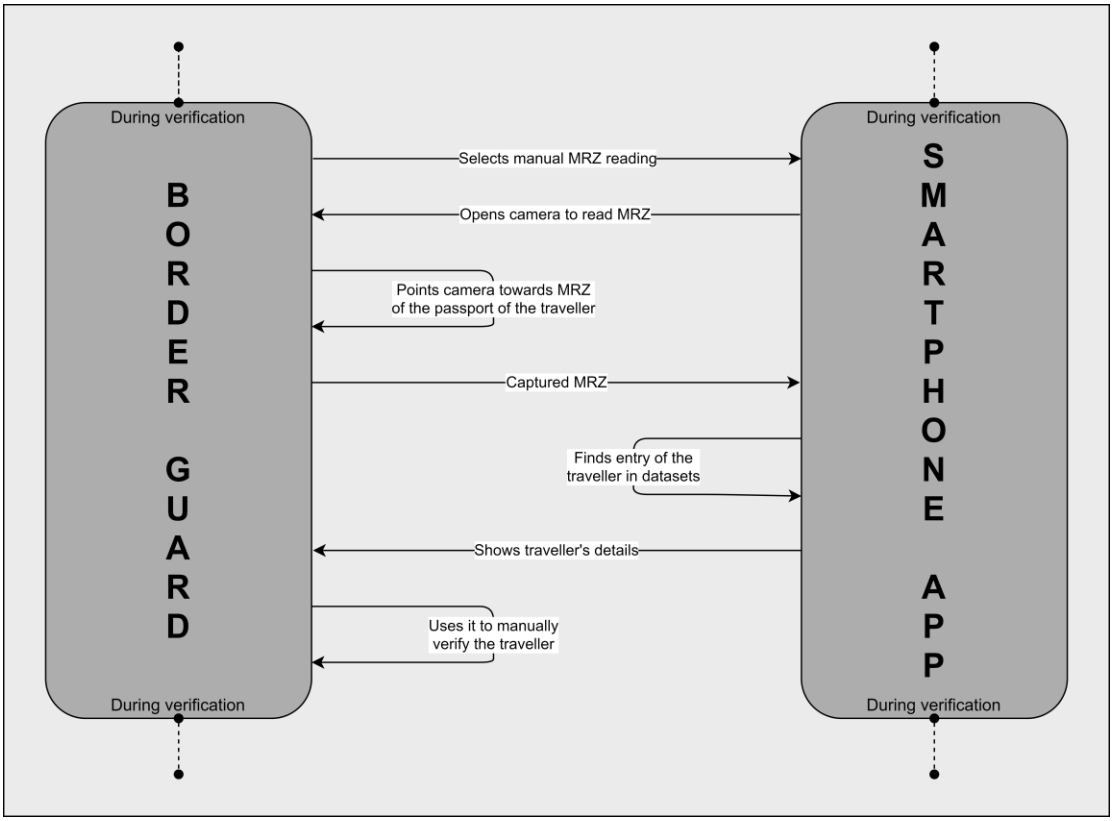

FIGURE 5: MANUAL PASSPORT CHECK (ONLY SUCCESS SCENARIO)

The app also provides a way to see the travellers that have finished verification (both success and failure cases). Each of these entries can again be viewed in detail. Also, at the end of the process, a summary can be seen where it shows all travellers that were successfully verified, all those who failed verification and all those who were expected on the coach but were not present.

If the border guard finds someone on board who has not enrolled before boarding the coach, the border guard checks that person manually either by stepping that person out of the coach and to a manual check point or manual checking on the spot.

3 ARCHITECTURE, COMPONENTS AND INTERFACES

This section describes the top-level architecture followed by external and internal components of smartphone application including the interfaces and the interaction of the smartphone application with the backend.

3.1 Top-level architecture

Figure 6 shows the top-level architecture. All three modules have already been briefly described in Section 2 above and the related steps in the processes from user point of view. In this section the third module, the smartphone application, will be described further in detail.

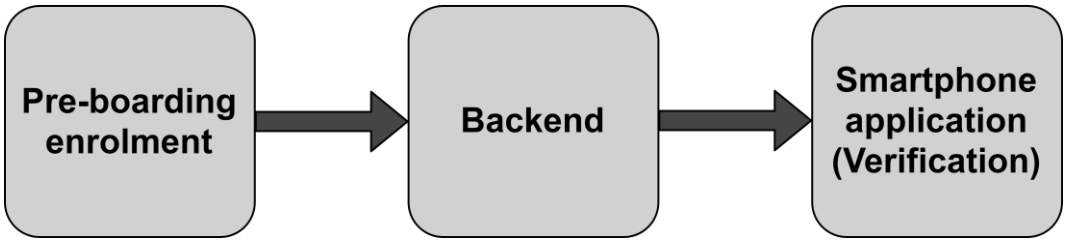

FIGURE 6: TOP LEVEL ARCHITECTURE

3.2 Smartphone application

The smartphone application plays the key role in the verification process that is undertaken by the border guard. The following are the key actions the smartphone application performs:

- Downloads travellers' enrolled and encrypted data sets from the backend server and stores it temporarily in the smartphone memory.
- Decrypts required data temporarily in volatile memory only at the time it is needed, thus avoiding permanent storage of un-encrypted data on the smartphone.
- Verifies travellers by taking a facial image and matching it with the images in the datasets.
- For travellers that fail facial image verification, the smartphone application provides an MRZ recognition feature through which the MRZ of the traveller's passport can be read and corresponding data can be fetched from the dataset for further verification.
- All data of a traveller can be viewed on the smartphone if the border guard needs to
- At the end of the process, the app shows a summary of the number of travellers verified, those not verified and those that were expected on coach but were not present.
- After the completion of verification process, all data is deleted.

Below (Figure 7) shows an overview of the components and interfaces of the smartphone application. The components can be divided into software and hardware components.

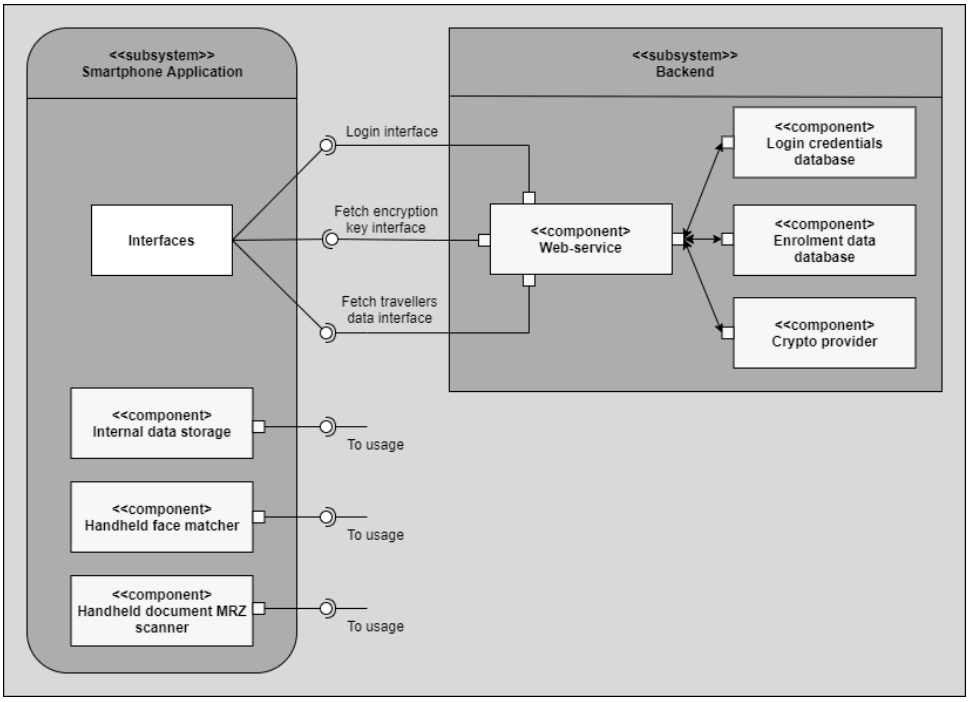

FIGURE 7: SMARTPHONE AND ITS SOFTWARE-HARDWARE COMPONENTS

3.3 Software components to smartphone

As shown in Figure 7, the software components comprise of backend databases, Crypto provider, backend server and internal data storage (on smartphone).

3.3.1 Backend databases

There are two separate databases set up. The first database stores login credentials of the border guard and second database stores the enrolled and encrypted data of travellers. For this project, the type of database that will be used is MongoDB [2][3].

The structure in the database will represent a JSON structure [4][5]. It will be a combination of JSON-key and value. The JSON-key will describe a particular field in the entire dataset and the value corresponding to the JSON-key will describe the actual data of that field, such that the dataset consists of key-value pairs.

3.3.1.1 Login credentials database

The smartphone application needs to be logged-in using login name and a password, before it can be used to protect it against unauthorised use. To securely store the login credentials for reference, a database on a backend server is used. An account is created for a border guard in this database by an administrator, by creating entries including a login name and a password.

The following data fields are defined for a login account (see also Figure 8) and will be stored in this login credentials database:

1. **Username:** The username of the

```
{
    username:    HASHED_VALUE,
    password:    HASHED_VALUE,
    token:       PLAIN_TEXT_VALUE,
    tokenTO:     PLAIN_TEXT_VALUE,
    isActive:    PLAIN_TEXT_VALUE
}
```

FIGURE 8: LOGIN CREDENTIALS DATABASE FIELDS

border guard.
2. **Password:** The password of the border guard.
3. **Token:** A token that is created during successful authentication and used to verify for any requests to enrolment data database and encryption key download.
4. **Token timeout:** The time when token has to be expired and border guard is forced to login again.
5. **Is account active:** Denotes whether the border guard is still allowed to use this account or not.

The username and password will not be stored in plain text but instead a hash over it (SHA256). The interface to this database and usage will be explained in detail in Section 3.5.1 "Login interface".

3.3.1.2 Enrolment data database

The smartphone application interacts with the enrolment data database to download the data set of travellers who have currently enrolled for a particular coach. A request is sent to the server hosting the enrolment data database to grab the data and a response is received that contains the actual data. This request and response takes place over a secured channel. All the data that is downloaded from database is encrypted. No data that is downloaded is in plain text or in decrypted format.

The following fields are foreseen to hold the enrolled data of a traveller. The structure of the dataset is based on the structure that is also proposed by ICAO and used in passports [6]. This structure has been taken into consideration because the traveller's biographic data is needed by the border guard during verification. This has been well described in the passport's data group structure by ICAO. Only relevant fields have been selected which should be sufficient for the border guard to identify and know important details of the traveller. The data structure is described below, and illustrated as it is written in the code in Figure 9:

1. **MRZ:** The MRZ of the document that the traveller uses to Enrol.
2. **Coach trip number:** The unique number of the specific coach ride.
3. **First name:** The first name of the traveller as read from the data group 1 (DG1) of the traveller's passport.
4. **Last name:** The last name of the traveller from DG1.
5. **Document type:** The type of document from DG1.
6. **Issuing state or organization:** The state or organization that issued the document from DG1.
7. **Sex:** The gender of the traveller from DG1.
8. **Nationality:** The nationality of the traveller from DG1.
9. **Passport face image:** The face image data DG2 from the passport.

```
{
    mrz:                    HASHED_VALUE,
    coachTripNumber:        PLAIN_TEXT_VALUE,
    firstName:              ENCRYPTED_VALUE,
    lastName:               ENCRYPTED_VALUE,
    docType:                ENCRYPTED_VALUE,
    issuingAuthority:       ENCRYPTED_VALUE,
    sex:                    ENCRYPTED_VALUE,
    nationality:            ENCRYPTED_VALUE,
    docFaceImage:           ENCRYPTED_VALUE,
    highResLiveFaceImage:   ENCRYPTED_VALUE,
    lowResLiveFaceImage:    ENCRYPTED_VALUE,
    faceTemplate:           ENCRYPTED_VALUE,
    backgroundCheck:        ENCRYPTED_VALUE
    encKey:                 ENCRYPTED_VALUE
}
```

FIGURE 9: ENROLMENT DATA DATABASE FIELDS

10. **High resolution face image:** The high resolution face image captured by the camera in the enrolment kiosk.
11. **Low resolution face image:** The low resolution face image captured by the camera in the enrolment kiosk.
12. **Face template:** A template created from the live high resolution face image that will be used for face matching during the verification process.
13. **Background check:** States whether the background check of traveller failed or passed.
14. **Enc key:** Used for cryptographic operations.

The interface to this database will be explained in detail in Section '3.5.3 Fetch travellers data interface'.

3.3.2 Crypto provider

As already mentioned, all data in the backend enrolment data database will be encrypted. The Crypto provider on the backend server will be generating and providing keys for the encryption of the data.

The handling of the encryption keys is described in Figure 10 below. The Crypto provider will generate a RSA key pair that will be used for encryption and decryption of the data. The public key is first downloaded by the KIOSK to encrypt the traveller's data before uploading the data set to the database. The smartphone application then downloads the private key that will be used for decryption on the smartphone.

A different key-pair is generated for every different coach journey. The Enrolment-KIOSK triggers the generation of key. The KIOSK asks the Crypto provider to make a new key-pair for a new coach trip number and this key-pair is then used throughout the enrolment and verification processes for this particular coach trip number. The module will also store this key-pair and provide it when needed to the border guard's smartphone.

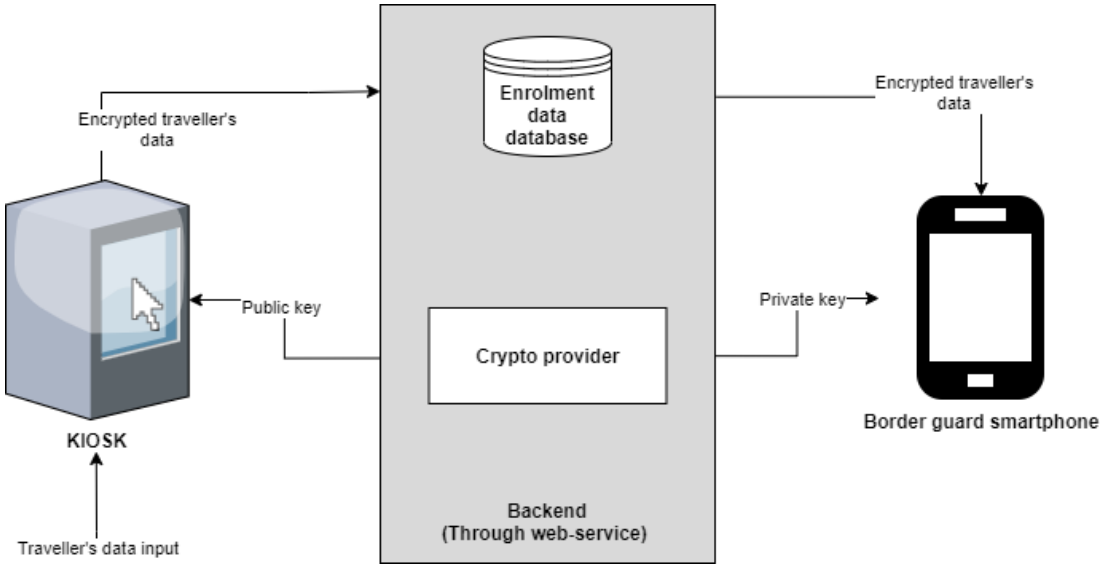

FIGURE 10: CRYPTO PROVIDER (BRIEF OVERVIEW)

The data in enrolment data database will be encrypted by the KIOSK. For this, a specific secret key (AES key) is generated that will be used to encrypt each fields in database. The encryption will be done on each value of the JSON-key in database and not on the JSON-key itself. As per section '3.3.1.2 Enrolment data database', the JSON-keys will remain un-

encrypted, while the values of each JSON-key will be encrypted. As shown in Figure 9, the first JSON-key “mrz” value will be a hashed value (SHA-256) over the actual MRZ of the document. The second JSON-key “coachIdentNum” will be in plain text and is only needed to know to which journey is this particular traveller enrolled for. All other JSON-key values will be encrypted by the secret key. The secret key will be then be encrypted by the public key received from backend and stored in “encKey” field.

3.3.3 Backend server

The databases and Crypto provider will be hosted on a backend server. For this project, for the development and prototype phase, the server will be hosted on Amazon Web Service [7][8].

The configuration of the backend will be done using the Node.js web-service. This is as shown in Figure 7. All external incoming requests will be handled by this Node.js web-service and internally used to work with the databases. Node.js has been selected due to its simplicity and speed by which it handles request-response [9]. All external requests are handled by this web-service which uses internal components to process those requests followed by returning responses as needed.

3.3.4 Internal data storage (on smartphone)

There are two types of data that are stored on the smartphone:

- A token that is generated during the authentication of the user of the smartphone application (border guard). This token is needed for authentication of download requests to the backend server for the verification process. This token is stored until the user logs out or until the application is uninstalled. When a new login is performed, a new token is generated and stored. This token is stored in a private memory of the application which means it cannot be accessed from anywhere outside the application.
- The downloaded enrolment data of travellers needed for verification process. This is stored only temporarily until the verification process runs and is deleted as soon as the verification process completes. The data is stored in an internal MongoDB Realm database and remains private to the application [10]. This means it cannot be accessed from anywhere outside but by only the intended application. All the data inside this database will remain in encrypted format.

3.4 Hardware components to smartphone

The hardware components comprises of handheld face matcher and handled document MRZ scanner. The hardware components are present in the smartphone.

3.4.1 Handheld face matcher

The matching of the face image, as captured using the smartphone camera, with the image from the data sets on the smartphone is done using a commercially available face matching technology from the company Neurotechnology [11]. The component of Neurotechnology used is called “Megamatcher” [12]. The performance data, as specified by the provider of the matching software is shown in Figure 11.

MegaMatcher 11.2 face engine specifications					
	Embedded / mobile (1) platform		PC-based (2) platform		Server platform
Template extraction components	Mobile Face Extractor	Mobile Face Client	Face Extractor	Face Client	Face Image Processing(3)
Template extraction speed (faces per minute)	45	50	45	100	3,000
Template matching components	Mobile Face Matcher	Mobile Fast Face Matcher	Face Matcher		Fast Face Matcher(2)
Template matching speed (faces per second)	3,000	200,000	40,000		200,000
Single face record size in a template (4) (bytes)	194 or 464 (configurable)				

FIGURE 11: MEGAMATCHER ENGINE SPECIFICATIONS [13]

On average, the smartphone should be capable of performing 3000 matches per second. This promises to be sufficient for a seamless face matching process during the verification process in the coach. The influence of lighting conditions and the practical use in such a scenario will be evaluated during further internal testing and also in the field tests. Also, how the smartphone's hardware specifications (such as camera, lens, processor, etc.) will play a role would be evaluated.

3.4.2 Handheld document MRZ scanner

The MRZ reading is performed using the MRZ reading SDK from Regula [14], who is a partner in the D4FLY consortium. The SDK will be integrated directly into the app. Using the smartphone camera along with the MRZ reading SDK, MRZ information from traveller's document will be read and extracted.

3.5 Interfaces to the smartphone application

The smartphone application handles three interfaces. The first interface interacts with the backend login credentials database for the authentication procedure during the login, the second interface interacts with the Crypto provider to download the private key for decryption, and the third interface interacts with the backend enrolment data database to download enrolled travellers' data.

As already described in Section '3.3.3 Backend server', all three interfaces are communicating with a web-service. A web-service is used to handle these requests to the databases on the backend. The channel between smartphone application and web-service will have a TLS encryption [15] and hence data transfer between them will be secure. An additional security mechanism is built in, where only those requests to the web-service are accepted, that are sent from a valid account of an authenticated border guard of the smartphone application.

3.5.1 Login interface

The smartphone application needs to be logged in with a valid border guard account before it can be used for verification process. The login credentials for a border guard are created and entered in the login credentials database. The hash value is calculated from the

credentials and the hash value is stored in the database. These credentials are then supplied to a border guard who can use it to login to the application. The smartphone then uses the login interface to authenticate a border guard and allow login (usage) of the application. The following diagram (Figure 12) depicts the communication during the login process, showing the "good case".

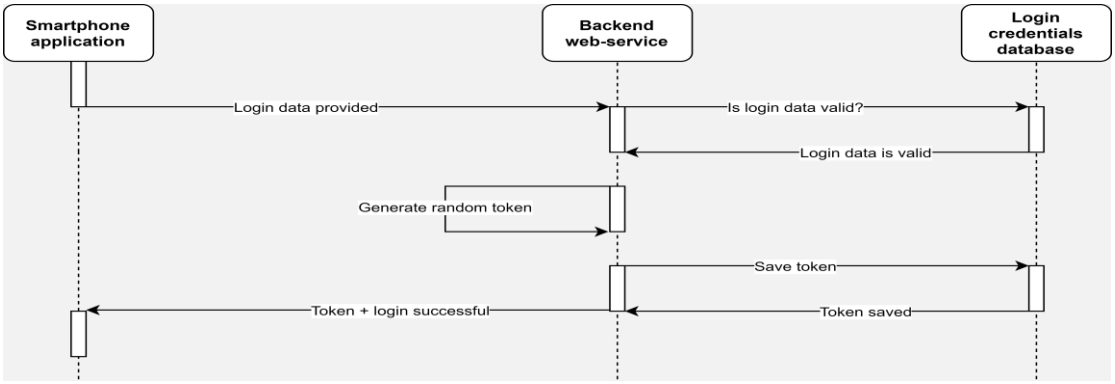

FIGURE 12: LOGIN INTERFACE

1. **Login data provided:** The username and password is sent to backend web-service for validation (only hashes of both are sent).
2. **Is login data valid?:** The credentials are checked in login credentials database whether they exist along with correct combination or not.
3. **Login data is valid:** The credentials have been found to be correct.
4. **Generate random token:** A random token, using Crypto of Node.js [16], is generated that will correspond to this login credentials. This token will now be needed for any further communication by this border guard with backend web-service.
5. **Save token:** The token is saved in login credentials database next to the actual credentials of this validated border guard.
6. **Token saved:** The token has been saved in login credentials database.
7. **Token + login successful:** A login successful response containing the token is sent to smartphone application.

3.5.2 Encryption key interface

The smartphone application, before beginning the verification process, requires the correct private key by which the data can be decrypted. To receive this key, a request is sent to the web-service where the request is validated based on the token supplied with the request. If the request is valid, the correct private key is sent in a response to the smartphone application based on the coach trip number provided. The following diagram (Figure 13) explains the process (only success case is shown).

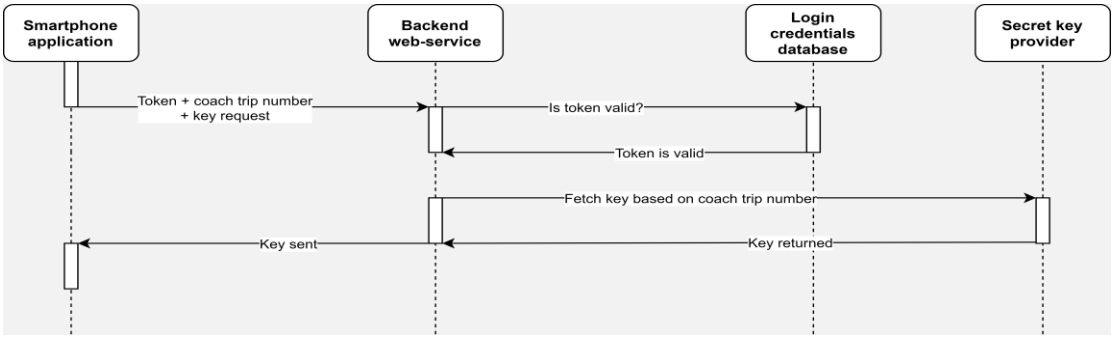

FIGURE 13: FETCH ENCRYPTION KEY INTERFACE

The description is as follows:

1. **Token + coach trip number + key request:** The token is sent to backend web-service for validation along with a request to get the private key. The coach trip number is also provided to let the backend know which particular key is required.
2. **Is token valid?:** The token is checked whether it is valid and not expired.
3. **Token is valid:** The token has been found to be correct and not yet expired.
4. **Fetch key based on coach trip number:** The key is being fetched from Crypto provider corresponding to the coach trip number provided.
5. **Key returned:** Crypto provider gives the key.
6. **Key sent:** The key is sent to smartphone application.

3.5.3 Fetch travellers data interface

This smartphone application interacts with the backend enrolment database to download the dataset of travellers who have currently enrolled for a particular coach. The data transfer is triggered by a request which is sent to the web-service and the web-service responds with the transmission of the data sets after a positive validation of the token, which is a part of the request.

In order to be able to start the verification process as quickly as possible, after the specific coach to be checked has been selected, and to avoid long waiting times, the application will download data based on priority order. By using this method, the verification process can be started already after the highest priority data is transferred. Depending on the quality and the performance of the mobile data connection of the user, the download times of the full database might vary.

In reference to Section '3.3.1.2 Enrolment data database' and Section '3.3.2 Crypto provider', the following priorities have been assigned to JSON-keys:

1. **Highest priority:** This category includes those JSON-keys that need to be downloaded first and foremost since the initial steps of verification process by border-guard relies on these JSON-keys. These JSON-keys are: "mrz" and "faceTemplate".
2. **Medium priority:** This category includes those JSON-keys that can start downloading after highest priority fields have finished downloading and even after the verification process has started. They are shown briefly after a traveller has been verified. These JSON-keys are: "firstName", "lastName", "lowResLiveFaceImage" and "backgroundCheck".
3. **Low priority:** This category includes those JSON-keys that can start downloading after medium priority fields have finished downloading. These include details that can be viewed if for a verified traveller further additional details need to be viewed by the border guard. These JSON-keys are: "docType", "issuingAuthority", "sex" and "nationality".
4. **No priority:** This category includes those JSON-keys that are not necessarily required in the process. These are usually not shown on the user interface of the smartphone application, but can be requested by the user and be displayed on demand. These JSON-keys are: "docFaceImage" and "highResLiveFaceImage".

The download of traveller's data will happen based on the priority order. A variation of lazy loading design pattern will be used to implement this interface [17]. The idea is to load only the most essential fields required (high priority fields) and then to start the verification process. This ensures the verification process starts as soon as possible avoiding long waiting

time. The verification process can already start while the other required fields (medium and low priority fields) will be downloaded in parallel to the verification process without interfering with the actual verification process. The fields that are not required in the standard process are not downloaded at all and will only be downloaded if explicitly needed or requested by the border guard. The following diagram (Figure 14) explains the process (only success case is shown).

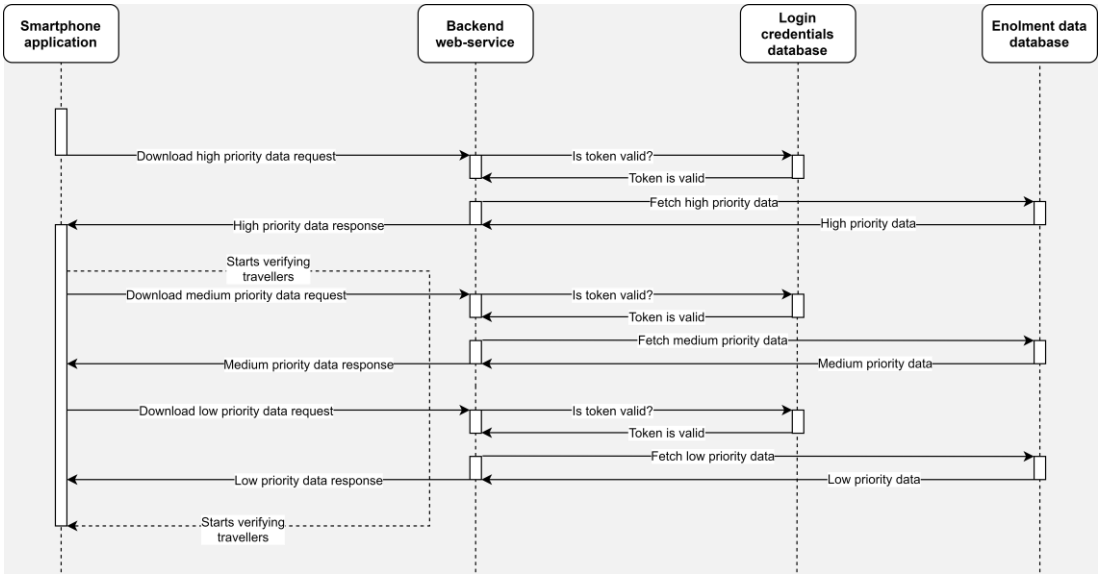

FIGURE 14: FETCH TRAVELLERS DATA INTERFACE

The description is as follows:

1. **Download high priority data request:** A request is made to the web-service to download the high priority JSON-key fields of all travellers identified by the coach trip number.
2. **Is token valid?**: The token is checked whether it is valid and not expired.
3. **Token is valid:** The token has been found to be correct and not yet expired.
4. **Fetch high priority data:** The web-service queries the enrolment data database to get all the data corresponding to high priority JSON-keys of all travellers identified by coach trip number.
5. **High priority data:** The data is received by the web-service.
6. **High priority data response:** This data is then forwarded to the smartphone application.
7. **Starts verifying travellers:** The smartphone application gets ready and the border guard can now start to verify the travellers.
8. **Similar steps from 2 to 7 are performed for other priority data requests in order: medium priority data request and then low priority data request**.

Steps 7 and 8 can happen in parallel in the background along with the verification process.

4 USER INTERFACE

The smartphone application is the client application that the border guard interacts with. Below (Figure 15) is a graphical description of all the screens that takes input or gives output to the border guard.

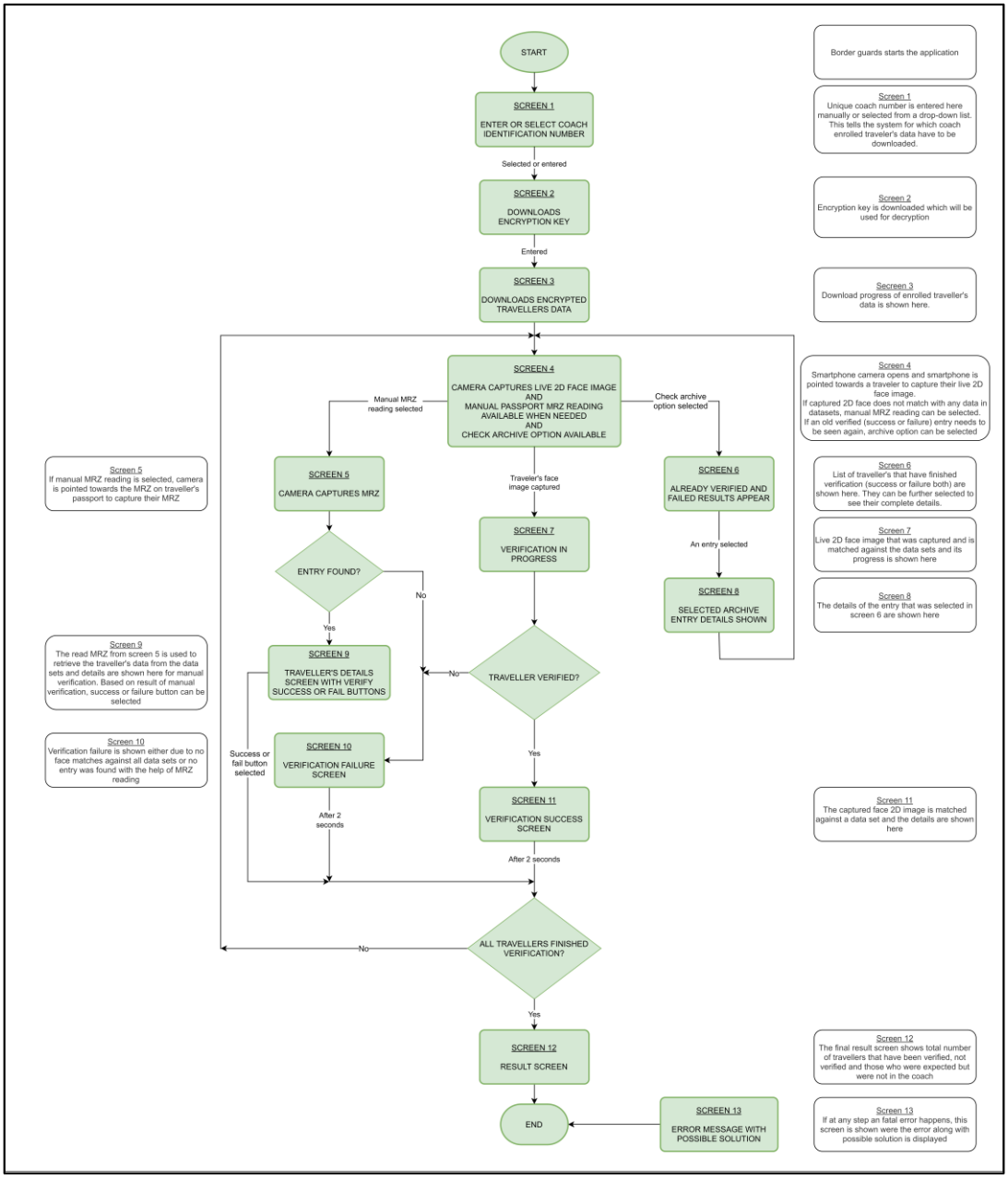

FIGURE 15: APP SCREENS' FLOW

The screens are interactive and assist in giving and taking information from the border guard. The interactive user procedure consists of the following steps (in regards to the screen):

1. **START:** The application starts.

2. **Enter coach trip number screen (Screen 1):** The border guard here enters the unique coach trip number which tells the app the travellers who are expected on this particular coach. The app can then use this information to download (only) the data of those travellers which have enrolled themselves for this particular coach.
3. **Download of encryption key screen (Screen 2):** The travellers' data that will be downloaded are all encrypted. To decrypt them, encryption key is needed. This key is downloaded here by the smartphone application.
4. **Download of Enrolled travellers' data (Screen 3):** The data of all travellers that have enrolled for the entered coach, are downloaded here. The screen shows the progress of the download.
5. **Main screen (Screen 4):** The main screen starts the camera of the smartphone to start grabbing faces of each traveller one by one. The camera of the smartphone has to be simply pointed towards a traveller and the app itself grabs the live 2D face image. This screen shows the camera feed to the border guard which can be used to identify a particular traveller.
    a. **Face matching (Screen 7):** Once a live 2D face image of a traveller is captured in screen 4, this screen 7 shows up to tell the border guard that face matching against all downloaded data sets is going on. It shows the progress in terms of total data sets checked.
    b. **Verification success screen (Screen 11):** If a match is obtained in screen 7, then this screen 11 shows up that shows the traveller has been successfully verified and along with this it also shows the travellers details (enrolled data).
    c. **Verification failure screen (Screen 10):** If no match is obtained in screen 7 against entire datasets, then this screen 10 shows up showing no match was found and manual check is needed. The border guard can then use additional manual check option available on screen 4 to perform manual check. This has been described below.

    The main screen also shows an option to manual check the traveller. This will be denoted by a button stating “perform manual check”. Selecting this button opens screen 5 (described below).

    a. **MRZ capture (Screen 5):** This screen opens the camera once again but this time to grab and read MRZ of a passport. The screen again shows camera feed which border guard can use to point towards a traveller’s passport data page. The app will automatically detect and read the MRZ and use it to fetch the data entry of this particular traveller in the datasets.
    b. **Traveller’s data details screen (Screen 9):** If an entry is found in datasets, this screen is shown which contains traveller’s data (enrolled data) which can be used to manually verify the traveller. The screen also shows two buttons: success and failure. If the manual verification is successful, then border guard can select success button, otherwise, border guard can select failure button to denote to the app that manual verification failed.
    c. **Data entry not found (Screen 10):** If no entry is found for this particular MRZ in data sets, this screen is shown where a messaging stating “traveller not enrolled” is shown.

    The main screen also shows an option to see/check the archive list. All travellers that have been verified previously can be seen again through this archive option. Selecting this option opens screen 6 (described below).

   a. **Archive list screen (Screen 6):** This screen shows a list of all previously verified travellers (both success and failure cases). The list will contain basic information such as traveller's picture, name, verification status, etc. Each entry in the list can be selected to see additional details in screen 8 (described below).
   b. **Selected traveller's details screen (Screen 8):** The details of the traveller's entry (that is selected in screen 6) is shown here. It includes all the enrolled data of the selected traveller.
6. **Result screen (Screen 12):** At the end of the process for a particular coach, a summary is shown. This screen shows that summary. It includes the total number of travellers verified on the coach, how many were successful and how many failed, how many needed manual verification, how many more travellers that had enrolled themselves but were not present on board, and how many travellers were present on board but had not enrolled themselves.
7. **Error screen (Screen 13):** If at any step a fatal error occurs due to which app has to shut down or restart, this screen will be shown. It contains the error message (what went wrong) and possible solution (what can be done to fix it).
8. **END:** The application ends.

5 DATA SECURITY CONSIDERATIONS

Data privacy, data protection and IT security are very important aspects, that have been considered during the design of the smartphone application. The measures and methods that are used in the smartphone application and the overall system configuration are briefly described in this section. The concept will be further analysed and potentially modified in the course of the project.

5.1 Pre-boarding enrolment

The necessary aspects that have been taken into consideration for pre-boarding enrolment are described below.

5.1.1 Encryption and data storage in the enrolment kiosk

The data, which is captured during the enrolment, mainly consists of the data that is read out of the traveller's passport along with the traveller's facial image taken by the kiosk, both of which are personal data of the traveller. In order to protect this data, it is encrypted using standard cryptographic encryption algorithms from open source software library functions before being uploaded to the backend enrolment data database.

Also, the dataset (which is generated in enrolment) always remains in runtime memory of the kiosk and is not stored temporarily anywhere. After the data set (of a particular traveller) is complete, it is immediately uploaded to the backend enrolment data database.

5.1.2 Data transmission from the kiosk to the backend

The traveller's enrolled data sets will be uploaded from kiosk to backend database over a secure TLS channel [15]. Also, the datasets will be uploaded in an encrypted format and not in plain text. This ensures the data always remain secured despite a secure TLS channel.

5.2 Backend

The necessary aspects that have been taken into consideration for the backend are described below.

5.2.1 Security added to server and databases

The data in databases will be secured on the server by username and password based authentication. To access the databases directly on backend server, a two-step authentication is required: first a valid username and password is required to login to the backend server, followed by a valid username and password to login to the databases. To access the databases through web-service, a valid request from a valid border guard account is required. A border guards' account is first validated with their username and password followed by generating tokens with expiring time. These tokens have to be sent to the web-service in order to be able to access the databases. The tokens also expire after every 2 days and need to be re-validated.

5.2.2 Data Storage in databases

The credentials stored in the login credentials database are hashed and not in plain text (Section '3.3.1.1 Login credentials database'). Also, the data stored in the enrolment data database is encrypted and not in plain text (Section '3.3.1.2 Enrolment data database'). The key used in encryption is also stored separately. Different keys are used for different coach trip numbers and hence data sets for different coach trip numbers cannot be decrypted with single key but respective key is needed.

5.3 Smartphone application

The necessary aspects that have been taken into consideration for smartphone application are described below.

5.3.1 App deployment and user authentication

As the application shall be used only by authorised personnel of the border guards, this application is not going to be distributed through standard public app stores, but will be downloadable from web-server through a download link. This link can only be accessed after positive verification with a dedicated username and password.

5.3.2 User authentication

In order to prevent unauthorised access to the application, a dedicated login mechanism is being implemented. It uses a dedicated login credentials database (Section '3.3.1.1 Login credentials database'), which stores the credentials of authorised personnel. This database is accessed by a dedicated admin, who can create, modify and delete accounts.

5.3.3 Data transfer from the backend to the smartphone

The traveller's enrolled data sets will be downloaded over a secure TLS channel. The data sets will be downloaded in encrypted format and not in plain text. This ensures the data always remain secured despite a secure TLS channel.

The decryption key will also be downloaded over a secure TLS channel.

Also, the web-service will allow the enrolled data or decryption key to be downloaded only after a valid authentication by the border guard's smartphone application account. This authentication is based on the tokens that have been described in Section '3.5.3 Fetch travellers data interface' and Section '5.2.2 Data Storage in databases'.

5.3.4 Data storage on the smartphone

The downloaded traveller's enrolled data sets will be stored temporarily on the smartphone in an internal database (Section '3.3.4 Internal data storage (on smartphone)'). This remains private to the application and cannot be accessed from outside. Also, all the data inside this database will remain encrypted. The key (to decrypt this data) will not be stored in internal database but will only remain in run-time memory. The data from this internal database will be fetched and decrypted (using the key) in runtime memory whenever it needs to be used. After it's usage, the decrypted value is deleted again. All this happens in runtime memory and not in internal database.

Also, the data in internal database will be deleted as soon as the verification process is completed.

6 CONCLUSION AND FUTURE WORK

Increasing the security and efficiency of border control processes is one of the main goals of the D4FLY project. Work package 6 in the project investigates options and alternatives specifically how smartphones could be used to verify traveller identities using pre-enrolled data.

A concept and implementation details of a smartphone application have been described, which is applied in a scenario, where border guards can execute the checking process in a confined space, such as in a coach. The smartphone application has the capability to download a pre-enrolled dataset, which relates to the passengers of a specific coach approaching the border. The smartphone camera is used to verify the travellers' identities and support the border guard perform checks before the travellers are allowed to cross the border.

Future work in D4FLY will include the following (will be reported in next version of the deliverable):

- Finalization of the smartphone application development
- Internal testing, focussing on the overall process
- Performing a field test, currently planned in M13/M14, to gather feedback for potential further improvements
- Implementation of improvements to create the final version of the smartphone application
- Final demonstration and evaluation of results

REFERENCES

[1] Grant Agreement-833704-D4FLY – Description of the action
[2] MongoDB benefits: https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm
[3] MongoDB benefits: https://docs.mongodb.com/manual/security/
[4] JSON: https://www.json.org/json-en.html
[5] JSON: https://www.w3schools.com/whatis/whatis_json.asp
[6] ICAO 9303: https://www.icao.int/publications/Documents/9303_p10_cons_en.pdf
[7] AWS benefits: https://www.tothenew.com/blog/top-10-benefits-of-using-aws/
[8] AWS benefits: https://aws.amazon.com/application-hosting/benefits/
[9] Node.js benefits: https://www.w3schools.com/nodejs/nodejs_intro.asp
[10] MongoDB Realm: https://www.mongodb.com/realm
[11] Neurotechnology: https://www.neurotechnology.com/
[12] Neurotechnology megamatcher: https://www.neurotechnology.com/megamatcher.html
[13] Neurotechnology megamatcher: https://www.neurotechnology.com/megamatcher-technical-specifications.html#face-engine
[14] Regula MRZ reading: https://mobile.regulaforensics.com/
[15] TLS: https://tools.ietf.org/html/rfc8446
[16] Crypto of Node.js: https://nodejs.org/api/crypto.html
[17] Lazy loading: https://developers.google.com/web/fundamentals/performance/lazy-loading-guidance/images-and-video

LIST OF FIGURES

ANNEX A: APP PACKAGES

The software architecture of the application is designed with the goal of ensuring maximum understandability and maintainability. Every package should only have its specific responsibility to fulfil and not know anything about what happens in other packages of the application. The packages communicate with each other only via interfaces (the only exception to this rule is "util", because this package contains multiple supporting methods/tools that are used frequently in many parts of the application).

Below (Figure 16) is the overview description of all packages involved.

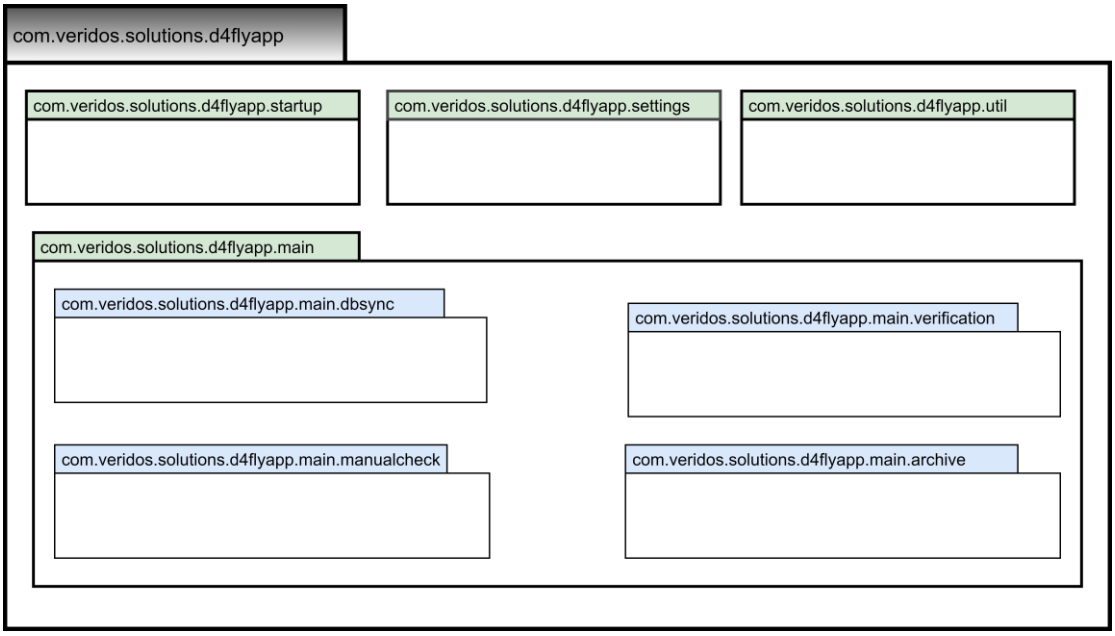

FIGURE 16: APP PACKAGES - OVERVIEW

Below each package that has been identified during the analysis phase is described.

I. Start-up package

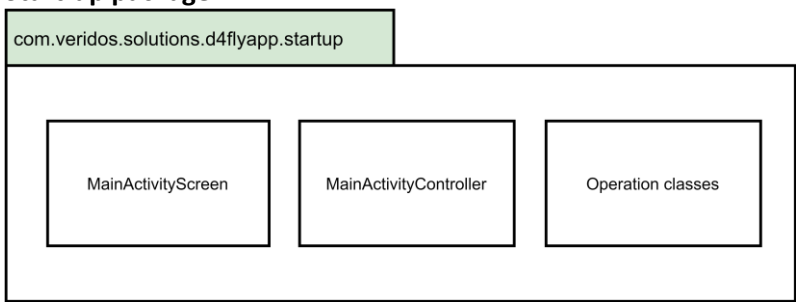

FIGURE 17: APP PACKAGES – START-UP

The start-up package (Figure 17) is the main start-up point of the application. The MainActivityScreen is the UI part of the package and MainActivityController is the controller class for the UI. The controller class in return calls different classes (denoted in general as operation classes) to achieve various functionalities. This package is responsible to initialize the app every time it is started fresh: ~~like~~ runtime

permissions, app updates, enabling (for example) location services or Bluetooth (based on needs), some decisions based on last execution states, etc.

II. Main package

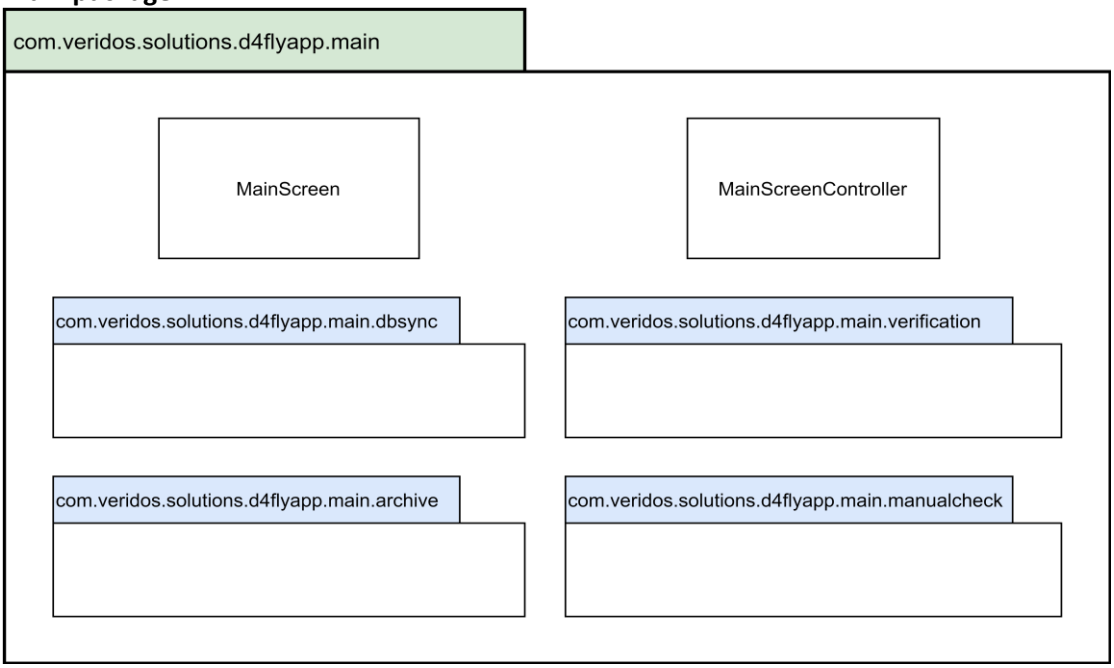

FIGURE 18: APP PACKAGES - MAIN

The main package (Figure 18) is used to control the main functionality of the app. It includes four sub-packages, namely dbsync, verification, manualcheck and archive. It shows main screen, represented by MainScreen, and controls the input border guard feeds it. The functionality is controlled by MainScreenController. When this screen starts, dbsync package is used to download all data of the travellers from backend database. The screen will then show the camera feed to capture face of a traveller. As soon as a face is captured, it is forwarded to be handled by verification package. In case the border guard wants to read MRZ of the traveller's passport, the camera is used to obtain MRZ of the passport. As soon as MRZ is read, it is forwarded to be handled by manualcheck package. If border guard wants to see the archive list of previously verified (both success and failure) travellers, control is forwarded to archive package.
As already mentioned, each sub-package is responsible for a certain task. This has been described below.

i. DB sync package

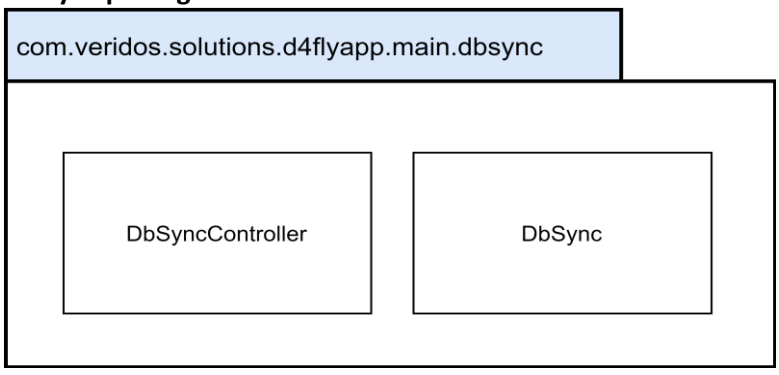

FIGURE 19: APP PACKAGES - DB SYNC

This package (Figure 19) is responsible for downloading the traveller's data from backend server database to local database on smartphone. It makes a secure connection to backend server, authenticates itself, fetches the required data, downloads them and enters it into local database. The DbSyncController is responsible for taking instructions from main package and giving result (control) back to it while DbSync class is called by DbSyncController to actually download the data.

ii. Verification package

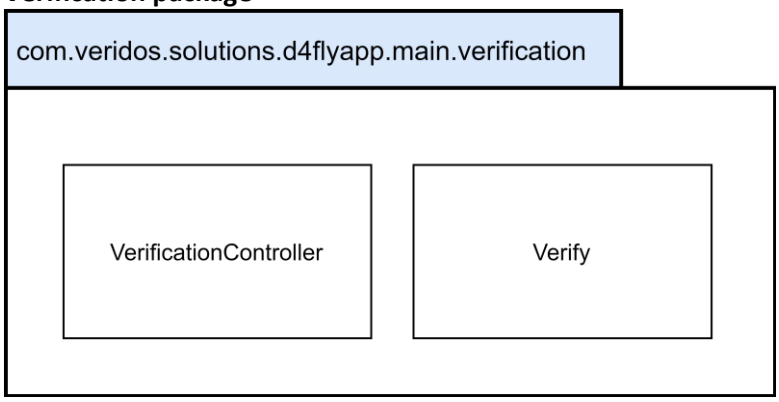

FIGURE 20: APP PACKAGES - VERIFICATION

This package (Figure 20) is responsible to verify whether the captured image of a traveller matches against any enrolled face present in the datasets. The VerificationController class is responsible for taking instructions from main package and giving result (control) back to it while Verify class is called by VerificationController to actually perform the face matching.

iii. Manual check package

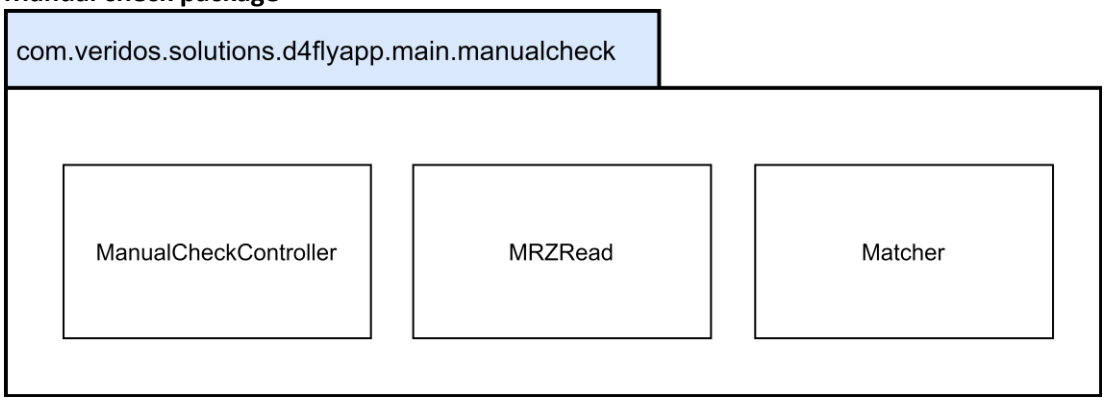

FIGURE 21: APP PACKAGES - MANUAL CHECK

This package (Figure 21) is responsible to assist the border guard in manual verification. When border guard wants to manually check the traveller, this package can be used. The ManualCheckController class is responsible for taking instructions from main package and giving result (control) back to it. The MRZRead class detects and reads the actual MRZ while Matcher class is used to find the entry set in the datasets and give the details to ManualCheckController to show it to the border guard for manual checking.

iv. Archive package

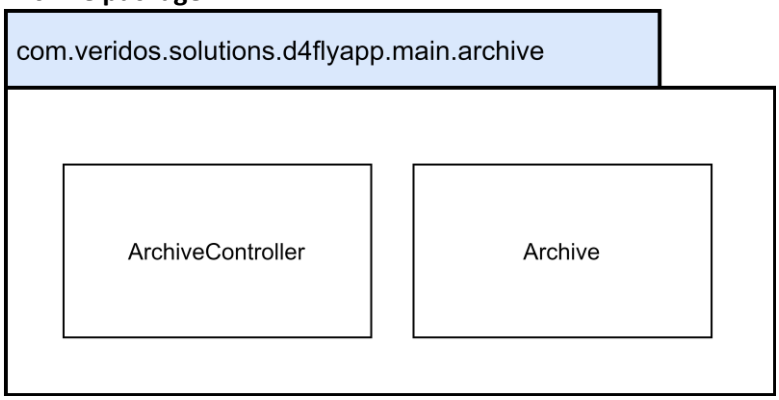

FIGURE 22: APP PACKAGES - ARCHIVE

This package (Figure 22) is responsible to show the archive list of already verified travellers (both success and failure cases). The ArchiveController class is responsible for taking instructions from main package and giving result (control) back to it. The Archive class fetches all the archive contents and gives them, along with travellers details, to ArchiveController to show it to the border guard.

III. Settings Package

This package is responsible for handling additional features for border guard such as language selection, showing app id, showing current app version number, etc.

IV. Util package

This package is responsible to assist in app development. It stores global constants, assists in managing app's internal memory, decryption of data and storing global static functions to be reusable at different points in different packages.