TED UNIVERSITY

PASS GUARD

SENIOR PROJECT II

LOW LEVEL DESIGN REPORT

Team:
Alp Eren Keskin
Doğuhan Cumaoğlu
Doruk Arslan
Hasanalp Temiz

Supervisor & Juries :
Gökçe Nur Yılmaz
Yücel Çimtay
İsmail Bora Çelikkale

Project Web Page URL:
Thepassguard.com

Contents

1.Introduction

1.1 Object Design Trading-offs

1. **Space vs Time:** The use of facial recognition technology and storing user data on the blockchain can take up a significant amount of space. However, it offers faster and more efficient authentication and data security. Therefore, the trade-off is to prioritize time and security over space. The Pass Guard application aims to provide a convenient and efficient solution for users to carry out various functionalities in their ecosystem from a single application. To achieve this goal, the application may require a larger storage space on users' devices. However, this may come at the cost of slower performance as larger storage space may require more time to access and retrieve data.

2. **Performance vs Security:** The use of NFC technology and facial recognition for authentication ensures fast and convenient transactions, but at the same time, it is crucial to ensure that these transactions are secure and protected from unauthorized access. The trade-off here is to balance performance and security to provide the best user experience while maintaining a high level of security. Pass Guard is designed with a facial recognition system to ensure the security of user data and transactions. However, this may affect the performance of the application, as facial recognition algorithms require computational power and may cause a delay in processing user requests. Therefore, there may be a trade-off between the performance and security aspects of the application.

3. **Portability vs Usability:** As a cross-platform mobile application, it is important to ensure that PASS GUARD is portable and easily accessible across different devices and platforms. However, it is equally important to ensure that the application is user-friendly and easy to use. The trade-off here is to balance portability and usability to provide a seamless user experience across all platforms. The Pass Guard application is expected to be released on all mobile platforms, which can increase its portability. However, this may come at the cost of usability, as the application may not be optimized for each platform and may require users to adapt to different user interfaces across different platforms.

4. **Existing components vs New components:** The use of existing components can speed up the development process and reduce costs. However, in some cases, new components might be required to ensure the functionality and security of the application. The trade-off here is to balance the use of existing and new components to provide a robust and secure application while keeping the development costs low. Pass Guard is designed to utilize its own token network and private coin for money transfers within the application. While this approach may offer more control and security over the payment system, it may require the development of new components and infrastructure, which can increase the complexity of the application.

5. **Reliability vs Supportability:** It is essential to ensure that the application is reliable and functions as intended, but at the same time, it is equally important to ensure that the application is supportable and easy to maintain. The trade-off here is to balance reliability and supportability to provide a stable and efficient application that can be easily maintained and updated. Pass Guard is developed with the goal of providing a reliable and secure solution for users. However, this may come at the cost of supportability, as more complex systems can be harder to maintain and troubleshoot. Therefore, there may be a trade-off between the reliability and supportability aspects of the application.

1.2 Interface Documentation Guidelines

The table below provides a preview of what the interface documentation will contain. A separate table will be used for each class to be presented, and the name of the relevant class will be placed at the top of the table. Under the class name, general definitions such as the description of the relevant class, its related features or functions will be available.

Moreover, after the related class description, the description of the methods to be found in the class will be explained under the name of Class Methods section. The design in this section will be made in such a way that the relevant method names and method responsibilities (and descriptions) will be found in the same row for each different method.

Each specified method name will be accompanied by parentheses '()' , return type and required parameters (e.g. , "public void myFunction(String myParameter)"). Similarly, a specific naming will be done for each function, each variable and each class name throughout the project (e.g., underscore namings for variables and camle cases for method names)

1.3 Engineering standards (e.g., UML and IEEE)

1. **Security standards:** The Pass Guard project needs to comply with various security standards to ensure the protection of user data and transactions. These standards include ISO/IEC 27001 for information security management, PCI DSS for payment card industry security, and GDPR for data protection and privacy.

2. **Mobile app development standards:** The Pass Guard mobile app needs to comply with various mobile app development standards, such as the Apple Human Interface Guidelines and Google Material Design Guidelines. These standards ensure that the app is user-friendly, easy to navigate, and follows best practices for mobile app development.

3. **Blockchain standards:** The Pass Guard project uses blockchain technology to store user data and transaction information. Therefore, the project needs to comply with blockchain standards, such as ERC-20 for token creation and management, and Hyperledger Fabric for private blockchain implementation.

4. **NFC standards:** The Pass Guard project uses NFC technology for payment transactions. Therefore, the project needs to comply with NFC standards, such as ISO/IEC 14443 and ISO/IEC 18092, which define the technical specifications for contactless smart card communication.

5. **QR code standards:** The Pass Guard project uses QR codes for entrance and attendance systems. Therefore, the project needs to comply with QR code standards, such as ISO/IEC 18004, which defines the technical specifications for QR codes.

6. **Web development standards:** The Pass Guard web application needs to comply with various web development standards, such as the W3C Web Content Accessibility Guidelines and OWASP Top 10 Security Risks. These standards ensure that the web application is accessible, user-friendly, and secure.

The concept UML diagram for the Pass Guard project may include several classes such as:

a. User: Represents the users of the Pass Guard application. It contains attributes such as name, ID, password, and facial recognition data.

b. PassCard: Represents the virtual card that users can use for payments and entry/exit to the ecosystem. It contains attributes such as card number, expiration date, and balance.

c. Transaction: Represents the transactions made by users in the ecosystem. It contains attributes such as transaction ID, date, amount, and status.

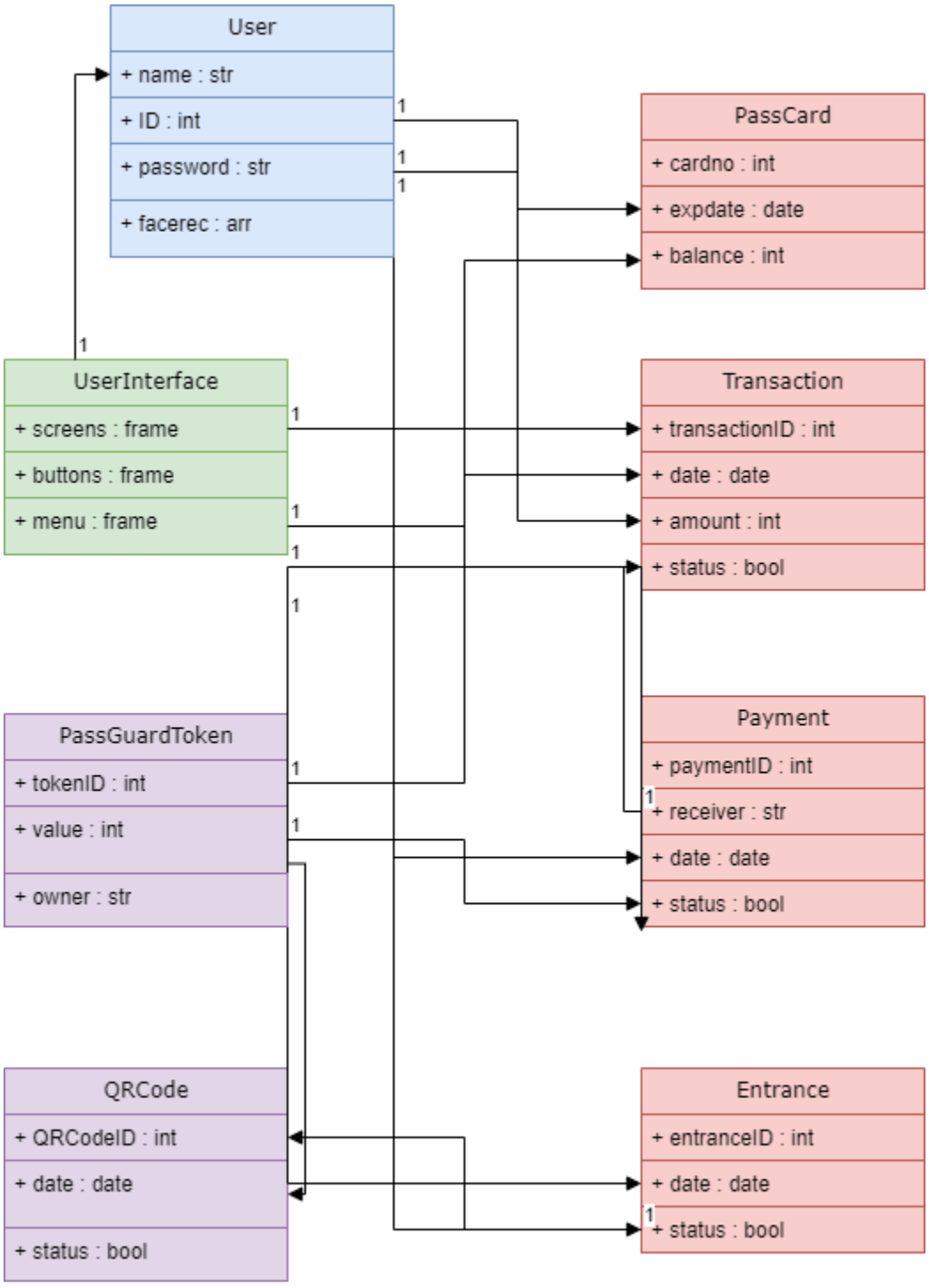
d. Payment: Represents the payments made by users in the ecosystem. It contains attributes such as payment ID, receiver, amount, and status.

e. Entrance: Represents the entrance/exit to the ecosystem by users. It contains attributes such as entrance ID, date, and status.

f. QRCode: Represents the QR codes used for entry/exit and attendance. It contains attributes such as code ID, date, and status.

g. UserInterface: Represents the graphical user interface of the Pass Guard application. It contains attributes such as screens, buttons, and menus.

h. PassGuardToken: Represents the token network used for money transfers within the Pass Guard application. It contains attributes such as token ID, value, and owner.


User
+ name : str
+ ID : int
+ password : str
+ facerec : arr
PassCard
+ cardno : int
+ expdate : date
+ balance : int
UserInterface
+ screens : frame
+ buttons : frame
+ menu : frame
Transaction
+ transactionID : int
+ date : date
+ amount : int
+ status : bool
PassGuardToken
+ tokenID : int
+ value : int
+ owner : str
Payment
+ paymentID : int
+ receiver : str
+ date : date
+ status : bool
QRCode
+ QRCodeID : int
+ date : date
+ status : bool
Entrance
+ entranceID : int
+ date : date
+ status : bool
1

1.4 Definitions, acronyms, and abbreviations

1.4.1 Definitions:

- PASS GUARD: The name of the mobile application and project that aims to carry users' profiles from ecosystems to the mobile application and ensure their verification and security through facial recognition.

- Facial recognition: A technology that uses biometric data to verify and identify individuals by analyzing and comparing their facial features.

- Ecosystem: Refers to the universities, large corporate institutions, or similar organizations that users can carry their business cards to the PASS GUARD application and perform various functionalities within the ecosystem.

- Blockchain system: A decentralized digital ledger that records transactions and data across a network of computers.

- Token system: A digital system in which a unique code, or "token," is generated and used to authenticate users or authorize transactions.

- Cross-platform: Refers to software that can run on multiple operating systems or devices, such as iOS and Android.

- Admin: Short for administrator, a person with special privileges and access rights to manage a system or application.

- Wep application: Refers to a web-based application, which is accessed and used through a web browser.

- Attendance system: A system used to track and monitor attendance, typically used in educational or business settings.

1.4.2 Acronyms and Abbreviations:

- NFC: Near Field Communication, a technology that allows two devices to communicate wirelessly when they are brought close to each other.
- QR: Quick Response, a type of two-dimensional barcode that can be scanned by a smartphone camera to quickly access information or perform actions.
- API: Application Programming Interface, a set of protocols and tools used to build software applications.
- UI: User Interface, the graphical layout and design of an application or website that users interact with.
- UX: User Experience, the overall experience that users have while interacting with an application or website.

2. Packages

This project is composed of two different packages: the Payment package and the Authentication package. The Payment package contains sub-packages related to payment functionalities, while the Authentication package contains sub-packages related to user authentication.

Payment Package

The Payment package is designed to allow users to make payments with the help of NFC technology. It consists of four sub-packages:

1. Payment Method: This sub-package provides users with the ability to manage their payment methods by adding or removing payment options that the application supports. It offers users the flexibility to customize their payment options based on their preferences and ensures a smooth payment experience.
2. Payment History: This sub-package is a key feature of the application that ensures users have complete visibility and transparency into their transactions. After a payment is completed, this sub-package displays a confirmation screen that includes detailed information about the transaction, such as the payment amount, payment method, date and time of the transaction, and any associated fees. This feature provides users with peace of

mind and confidence that their payment was successful, and they can easily track and manage their payment history within the application. In summary, the Confirmation Screen sub-package is a valuable feature that enhances the overall user experience and builds trust in the application's payment system.

3. Payment Confirmation: This sub-package provides users with a confirmation screen after completing a payment, displaying the details of the transaction.

4. Payment Processing: This sub-package handles the processing of payments, including communicating with payment gateways and updating the user's payment history. In addition to processing payments, the Payment Processing sub-package also ensures the security and confidentiality of all payment-related data. It utilizes encryption algorithms and secure communication protocols to protect sensitive information such as credit card details and transaction history.

Authentication Package

The Authentication package is designed to provide secure user authentication for the mobile application. It consists of three sub-packages:

1. Login: This sub-package is a fundamental component of the application that enables users to log in securely and access their account information. This sub-package provides users with the ability to create a unique username and password combination, which is encrypted and stored securely within the application.

2. Registration: This sub-package allows users to register for the application by providing their personal information and creating a new account.

3. Security: This sub-package is a critical component of the application that ensures the protection of user information. It includes advanced security measures such as password hashing and two-factor authentication to safeguard users' data from potential security breaches. Additionally, this sub-package provides users with the ability to customize their security settings based on their preferences, enabling them to take an active role in securing their information. Overall, the Security sub-package is an essential component of the application that promotes user trust and confidence in the platform's security.

3. Class Interface

3.1 Admin Application

Authorization Class	
This class is responsible for checking shared preferences (local admin data), navigating the onboarding pages when needed, and operating login and register processes.	
Class Methods	
Public boolean signUp():	Compares the provided admin credentials by the regexes. If it is successfull, allows admin to register to system by assigning a token to shared preferences
public void showOnboardings():	By considering the local preferences, it understands wheter the app installed for the first time or not. If it is the first time, it will navigate admin to the onboarding views.
public void cantSignIn():	If the admin has forgotten his current password, it helps to reset his password by means of the link sent to the e-mail address on the system.

Admin Homepage	
This class is responsible for displaying and updating information about admin users. It can access and view information about users and update or delete it if necessary.	
Class Methods	
Public boolean getUser()	Admin can access user information with this function (username, password status)
Public boolean disableUser():	Admin can disable the user's account with this function. If he uses the function again, the user becomes active again.
Public boolean getUserBalance():	With this function, the admin can monitor the user's account status (how much money they have, recent transactions, and purchases).
Public boolean getUserVector():	Admin can monitor the user's face vector with this function.

3.2 Mobile Application

Authorization Class	
This class is responsible for checking shared preferences (local user data), navigating the onboarding pages when needed, and operating login and register processes.	
Class Methods	
Public boolean signIn():	Verifys the provided user credentials and allows users to login to the screen by assigning a token to shared preferences.
Public boolean signUp():	Compares the provided user credentials by the regexes. If it is successfull, allows user to register to system by assigning a token to shared preferences
public void showOnboardings():	By considering the local preferences, it understands that wheter the app installed for the first time or not. If it is the first time, it will navigate user to the onboarding views.
public void cantSignIn():	If the user has forgotten his current password, it helps to reset his password by means of the link sent to the e-mail address on the system.

Home Class	
With this class, the user can view the information he/she has. Since it will be designed as the base class of the application, all the relevant redirections will take place within this class. With the help of this class, the announcements and notifications to be presented to the users within the application ecosystem are also provided.	
Class Methods	
Public List<String> getAnnouncements ():	Checks the server for the recent announcements, if any assigned to user, it is responsible for notifiying the user

Public void handleBottomBarNavigation (Page view):	As it is the base class of the application, it is responsible to make correct navigations by considering the correct animations. User will be able to navigate diverse views by trigerring that function. All the pages should be stored as enum in the class and to be used inside the application.
public CardModel getUserCard():	There should be a quick navigation for the user card via a pop up view. This function is responsible for getting the card data and present it to user.
public void handleState():	It is responsible for evaluating the requests to be made throughout the application and setting the application's state correctly depending on the changes by calling the relevant functions.

User Class	
It is the responsibility of this class to display and update the information found about the user. Users can access and view relevant information and update or delete if necessary.	
Class Methods	
Public UserModel getUserInfo ():	It is responsible for returning the data of the user such as name, surname, card number, account information and presenting them on the relevant widget.
Public boolean updateUserInfo (UserModel user):	It is responsible for updating the information owned by the user, if requested. It compares the changed information with the current information and accordingly creates a new user model. A request is made for the created model and the user information is updated. The result of the change is returned with a boolean value.
public boolean logOut():	It allows the active user in the system to log out and clears all currently held information. The result of the operation is returned with a boolean value.
public boolean deleteAccount():	It is responsible for deleting the account in the system if the user requests it. The result of the operation is returned with a boolean value.

Scan Class	
QR code operations within the application are under the responsibility of this class. Validation and recognitions on QR codes take place in this class.	
Class Methods	
public boolean validateQRData():	It validates the created qr code and confirms whether it conforms to the format used in the application. A booelan value is returned as a result of the operation.
public QRModel getQRData():	After the current user is verified, the information in the validated qr code is returned to the user. A QRModel is created representing the fields related to the information obtained and this model is returned to the user as a result of the operation.
public void generateQRData(QRModel model):	With the related function, users can embed the data of this model in the newly created QR code by sending a new QRModel as an argument with the information they provide.

Enterance Class	
It allows the user to enter the relevant area from specified points using the virtual card view. Before using the virtual card, the user must be validated and then the id of the relevant domain must be determined using the NFC feature. If the NFC feature is not active on the platform used, the user can use the existing qrCode.	
Class Methods	
public int scanCurrentPoint():	It uses the nfc protocol to detect the area where the user is located and obtains the id of the relevant area. In this way, the request to be made to the server by the application is evaluated and the desired door is opened by returning the placeID as an int value. If the nfc feature is not available to the users, they can obtain the same data by reading the qr code to be found at the entry point with the help of Scan Class.

public boolean validateUser():	It is used to validate the user before the actions to be taken. In this process, the user is validated by using the models in the faceRecognition class. Depending on the function result, the request is continued or canceled. This result is represented by returning a boolean value.
public void enterWithCard(int fieldID):	After the user is validated and the id of the relevant field is determined, the relevant field is given as a parameter and the operation related to the function help is performed for the active user. After the transaction to the server is successful, the relevant door engines are triggered by the server and allow user to enter.

Transaction Class	
It is responsible for the control and provision of transactions that users will make within the ecosystem. Relevant transactions can be realized over the NFC protocol at various payment points, as well as over qr code on devices where the NFC feature is not active.	
Class Methods	
public boolean placeTransaction(TransactionModel model):	The relevant function is responsible for performing the transaction after the users are validated. The transaction can take place at the payment point or between two users. This function is triggered automatically after getTargetByQR() or getTargetByNFC() execution. Result returned with a boolean value.
public boolean validateUser():	It is used to validate the user before the actions to be taken. In this process, the user is validated by using the models in the faceRecognition class. Depending on the function result, the transaction request is continued or canceled. This result is represented by returning a boolean value from function.
	If the user wants to perform the operation with the help of qr code, this function is triggered. The QR code belonging to the target is evaluated (it can be between payment points or users) and the target

public TransactionModel getTargetByQR():	id is obtained from the relevant result. Similarly, if an "amount" value is entered in the scanned qr code, it is presented to the user and asked to confirm, otherwise the user is presented with an input field to specify an amount and the obtained data is returned as the Transaction model
public TransactionModel getTargetByNFC():	If the user wants to perform the operation with the help of nfc scan, this function is triggered. The id of the relevant payment point is returned from the value currently read. If there is an amount value in the value read, it is automatically assigned and the user is asked to confirm this amount. Otherwise, the user is asked the amount they want to enter with an input screen and the obtained data is returned as the Transaction model.

FaceRecognition Class	
It is responsible for real-time face scanning for validation of the user and to return a response by comparing it with the scanned face data specific to the registered user in the system.	
Class Methods	
public faceScanModel scanFaceRealTime():	The user's face is analyzed from various angles in real time by using the camera sensor of the device over the mobile operating system used, and it is turned into a map that can be analyzed. The result obtained is returned as a faceScanModel as a result of the operation.
public boolean validateFace():	It compares the user-specific scanned face data entered on the system with the user's real-time face scan. In this process, it uses the scanFaceRealTime() function and the user's scanned data stored in the system. The result of the operation is returned with a boolean value.

4.Glossary:

1. Pass Guard: The name of the mobile application designed to carry user profiles from various ecosystems and to ensure the verification and security of users through the application's facial recognition system.

2. Ecosystem: A term used to describe a particular community or organization within which Pass Guard is used, such as a university, corporate institution, or similar entity

3. Facial recognition system: A technology used to verify the identity of users by analyzing their facial features.

4. Blockchain system: A distributed database that stores data in a secure and decentralized manner.

5. Token system: A system that utilizes tokens to represent a specific asset or value in a blockchain network.

6. NFC technology: Near Field Communication technology that enables contactless communication between devices.

7. QR code: A type of two-dimensional barcode that can be scanned using a mobile device's camera.

8. Transactions section: A section of the Pass Guard mobile application where users can make payments within the ecosystem they are in.

9. Entrance section: A section of the Pass Guard mobile application that allows users to easily log in or log out of an ecosystem by verifying themselves through the facial recognition system.

10. QR code scanner page: A page of the Pass Guard mobile application used for scanning QR codes to perform various functionalities.

11. Web application: A type of application that runs on a web browser and is designed to be accessed through the internet.

12. Performance: A measure of how well a system or application functions in terms of speed, reliability, and efficiency.

13. Security: A measure of the level of protection provided to data, systems, and users against unauthorized access, theft, or damage.

14. Usability: A measure of how easy and intuitive an application is to use for its intended users.

15. Portability: A measure of how easily an application can be transferred or used across different platforms or devices.

16. Reliability: A measure of the consistency and dependability of an application in performing its intended functions.

17. Supportability: A measure of how easily an application can be maintained, updated, and supported over time.

5.References:

1. *https://towardsdatascience.com/6-best-projects-for-image-processing-with-useful-resources-f860f3dfe977*

2. *https://www.pantechelearning.com/product/matlab-code-for-segmentation-based-on-skin-colour/*

3. *https://www.pantechelearning.com/product/matlab-code-for-image-encryption/*

4. *https://www.pantechelearning.com/product/matlab-code-for-head-pose-recognition/*

5. *https://www.pantechelearning.com/product/gender-recognition-using-image-processing/*

6. *https://www.pantechelearning.com/product/matlab-code-for-depth-estimation-using-image-processing/*

7. *ElSaddik,A.(2018).Digitaltwins:Theconvergenceofmultimediatechnologies.IEEE multimedia, 25(2), 87-92.*

8. *Brooks, F. P. (1999). What's real about virtual reality?. IEEE Computer graphics and applications, 19(6), 16-27.*

9. *Tourlakis, G. J. 2003]. Lectures in Logic and Set Theory, Cambridge University Press,*
10. *Cambridge. UK.*

11. *Toussaint, G. T. [1982). "Computational Geometric Problems in Pattern Recognition," In Pattern Recognition Theory and Applications, Kittler, J., Fu, K. S, and Pau, L. F.*
12. *(eds.), Reidel, New York. pp. 73-91.*

13. *Tsai, J.-C., Hsich. C.-H, and Hsu, T.-C. (2000]. "A New Dynamic Finite-State Vector Quantization Algorithm for Image*
14. *Compression," IEEE Trans. Image Processing, vol. 9, no. 11, pp. 1825. 1836.*

15. *Tsujii, O., Freedman, M. T., and Mun, K. S. (1998]. "Anatomic Region-Based Dynamie*

16. Range Compression for Chest Radiographs Using Warping Transformation of Cor-related Distribution," /EEE Trans. Medical Imaging, vol. 17, no. 3. pp. 407-418.

17. Udpikar, V. R. and Raina, J. P. (1987). "BTC Image Coding Using Vector Quantization,"
18. JEEE Trans. Comm., vol. COM-35, no. 3, pp. 352-356, Ueda, N. [2000]. "Optimal Linear Combination of Neural Networks for Improving Classification Performance,"' IEEE Trans. Pauern Anal. Machine Intell., vol. 22, no. 2,
19. pp. 207-215.

*20. Ullman, S. [1981]. "Analysis of Visual Motion by Biological and Computer Systems,***
21. IEEE Computer, vol. 14, no. 8, pp. 57-69.

22. Umbaugh, S. E. (2005]. Computer Imaging: Digital Image Analysis and Processing, CRC
23. Press, Boca Raton, FL.

24. Umeyama, S. [1988]. "An Eigendecomposition Approach to Weighted Graph Matching
25. Problems," IEEE Trans Pattern Anal. Machine Intell., vol. 10, no. 5, pp. 695-703.

26. Unser, M. [1995]. "Texture Classification and Segmentation Using Wavelet Frames,"
27. IEEE Trans. on Image Processing, vol. 4, no. 11, pp. 1549-1560.

28. Unser, M., Aldroubi, A., and Eden, M. [1993]. "A Family of Polynomial Spline Wavelet
29. Transforms," Signal Proc., vol. 30, no. 2, pp. 141-162.

30. Unser, M., Aldroubi, A., and Eden, M. [1993). "B-Spline Signal Processing, Parts I and
31. II,"' IEEE Trans. Signal Proc., vol. 41, no. 2, pp. 821-848.

32. Unser, M.. Aldroubi, A., and Eden, M. (1995]. "Enlargement or Reduction of Digital Images with Minimum Loss of Information," IEEE Trans. Image Processing, vol. 4,
33. no. 5, pp. 247-257.

34. Vaidyanathan, P. P. and Hoang, P. O. [1988]. "Lattice Structures for Optimal Design and
35. Robust Implementaion of Two-Channel Perfect Reconstruction Filter Banks," JEEE
36. Trans Acoust., Speech, and Signal Proc., vol. 36, no. 1, pp. 81-94.