Daffodil International University

Department of Computer Science and Engineering

SRS REPORT

Assignment No.: 05

Assignment on: Software Requirements Specification (SRS)

Submitted By :

Rabbi Hassan

ID:221-15-5570

Section:61-U

CSE236 | SOFTWARE PROJECT - 2

Submitted to: Md. Golam Rabbany

Dept. of CSE, Daffodil International University
Ashif Hasan
Manager
Airline Flight Booking System
Dhaka 1204

Subject: Application for agreement of Airline Flight Booking System Project.

Dear Mr sarker
I hope you are well. I am writing to propose the development of a comprehensive Airline Flight Booking System that I believe will significantly enhance the efficiency and quality of our airline flight service. I am the member of SoftTech company. I have identified the hole system of your airlines company and decided to create a unique Airline Flight Booking System. Thank you for considering this proposal. I am excited about the potential this project holds for our organization and look forward to the opportunity to discuss it in more details.

Sincerely
Rabbi Hassan
Head
Application development Brach
SoftTech

Table of Contents

Software Requirements Specification for Airline Flight Booking System

1.Introduction

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for an Airline Flight Booking System. The system aims to provide a user-friendly platform for passengers to search, book, and manage their airline flight reservations.

2.Overall Description

The Airline Flight Booking System is a web-based application that enables passengers to easily search for and book flights, manage their reservations, and view their travel history. The system provides a comprehensive suite of features to cater to the needs of both regular and occasional travelers.

3.Functional Requirements

3.1 Search and Booking

- Users can search for flights by specifying their desired origin, destination, and travel dates.
- The system displays a list of available flights, including departure and arrival times, airlines, and fares.
- Users can select the desired flight and provide passenger information, including names, contact details, and passport numbers.
- The system validates passenger information and displays the total fare, including taxes and fees.
- Users can proceed to the payment gateway to complete the booking process.

3.2 Payment Processing

- The system integrates with a secure payment gateway to process credit card transactions.
- Users enter their credit card information and confirm the payment.
- Upon successful payment, the system generates and sends an electronic ticket (eticket) to the user's email address.

3.3 Reservation Management

- Users can view their upcoming and past reservations.
- Users can cancel or modify their reservations, subject to airline policies and fees.
  The system sends email notifications for booking confirmations, cancellations, and modifications.

3.4 User Management

- Users can create an account to store their personal information and preferences.
  Users can manage their profile information, including contact details and passport details.
- Users can reset their passwords if they forget them.

4. Non-Functional Requirements

4.1 Performance

- The system should respond to user requests promptly and efficiently.
- The system should handle multiple concurrent users without performance degradation.
- The system should be able to process large volumes of data seamlessly.

4.2 Security

- The system should implement robust security measures to protect user data.
- All data transmissions should be encrypted using secure protocols.
- The system should adhere to industry-standard security practices and regularly undergo security audits.

4.3 Usability

- The system should have a user-friendly interface that is easy to navigate and understand.
- The system should provide clear instructions and guidance throughout the booking process.
- The system should be accessible to users with disabilities.

4.4 Reliability

- The system should be highly reliable and available to users 24/7.
- The system should have a robust disaster recovery plan in place.
- The system should undergo regular maintenance and updates to ensure its stability and performance.

5.Login

Description: This function allows a registered user to login his

account using his

frequent flyer number with the airline and password. If a user is not registered, the website shall allow the user to enroll first. The system will check both the frequent flight number and password, when a user attempts to login

6.Enrollment

Description:

account with the website. In order to create a new account, the user has to provide required information such as first name, last name, email address and password. Other optional information, such as phone number, credit card information and mailing address, can be provided during the registration process

The system checks if all required data are provided and then will prompt the user to enter additional information, if required. After all required information is provided, the system auto-generates a unique frequent flyer number that the user must use as username for future authentications. The system shall auto-generate this number in less than five seconds

.

7.Book Flights

Description:

The user can use the Reserve Seat function to reserve seats for an airplane flight. The seats to be reserved are initially found through the user's previous bookings. These bookings were previously completed through the book flight function previously completed The system shall display available seats for the departing and returning flights booked by the user. The user selects seats from each flight, where the number of selected seats from each flight is the number that the user booked on that particular flight. Once the flight seats are selected, the user confirms the seat selection

8.Flight Status

Description:

1.A flight number and Date OR

2.Departing/Arriving Cities and Date.

The system will display matching flight information including the following fields:

- Flight Number
- Departure City
- Arrival City

9.Flight Schedules

Description:

This section of the system shall allow a user to query flight schedules based upon simple input criteria. The user will provide departure and arrival cities, and a departure/return date. If any flights match the criteria, the system will display the following information:

- Flight Number
- Departing City & Date/Time
- Arriving City & Date/Time
- Number of Available Seats

The system shall define a “matching” flight as one that uses the departure/arrival cities at a flight time greater or equal to the time provided by the user. Otherwise, the system shall alert the user that no matching flights can be found.

10.My Account

Description:
This section gives the user the power to view, save, edit or delete the information stored in his/her account. The user can check his/her accumulated points, look at the status of a flight that was booked, cancel a flight that was already booked (optional) and change his/her address, phone number, email or password. This feature is not available for nonregistered user.

11.Logout

Description:

The Logout section provides a way for the user to securely log out of the system. This process will save all user operations when he/she exits the system. If a user wishes to continue accessing the website, he/she must log-in again to access user features.

12.User Classes and Characteristics

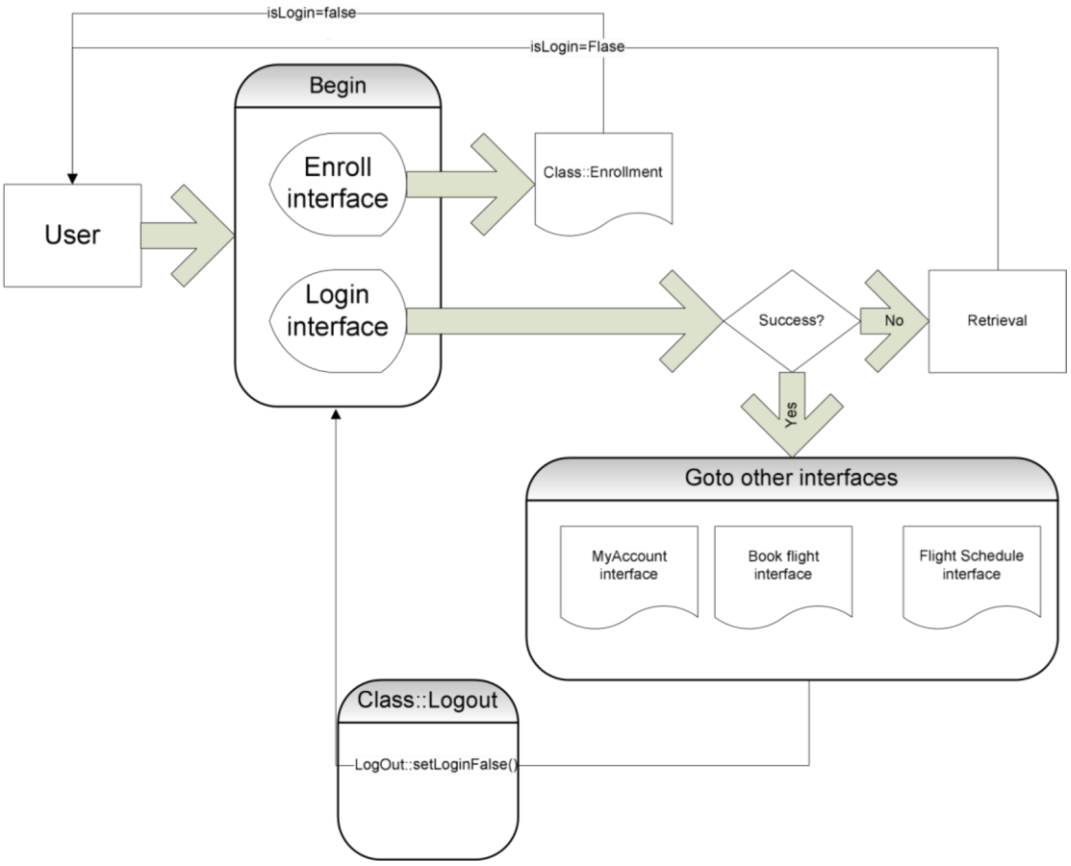
The main actors in the system are the user, a flight and a Flight Seat. The user will select a flight and book seats on the flight. They will then reserve specific seats on that flight. Brief descriptions of these classes follow:

- User Has properties like Name, Address, Age Associated with Flight Miles accumulated and Credit Card information.
- Flight Has properties like Departing/Arriving City, Departure/Arrival dates and times, Miles, and an identifying Flight Number.
- Flight Seat Has properties of identifying seat number, reserved and flight number Associated to Flight by flight number

Our user may be associated with multiple flights, and many users may be associated with any particular flight. Thus, a many-to-many relationship exists through the act of booking flights. Flights are associated to many Flight Seats. Each Flight Seat is only attached to one Flight. So, this is a one-to-many relationship.

The flight is but an object to be acted upon, so careful emphasis should be placed on satisfying the user in his/her booking experience. The user is our primary customer.

13.Use Case:

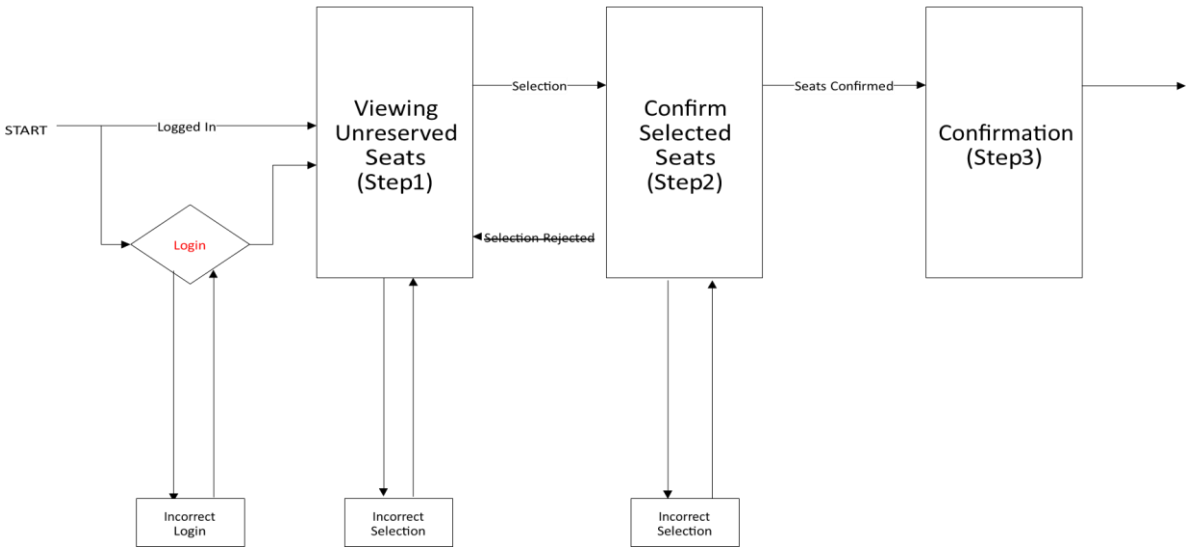


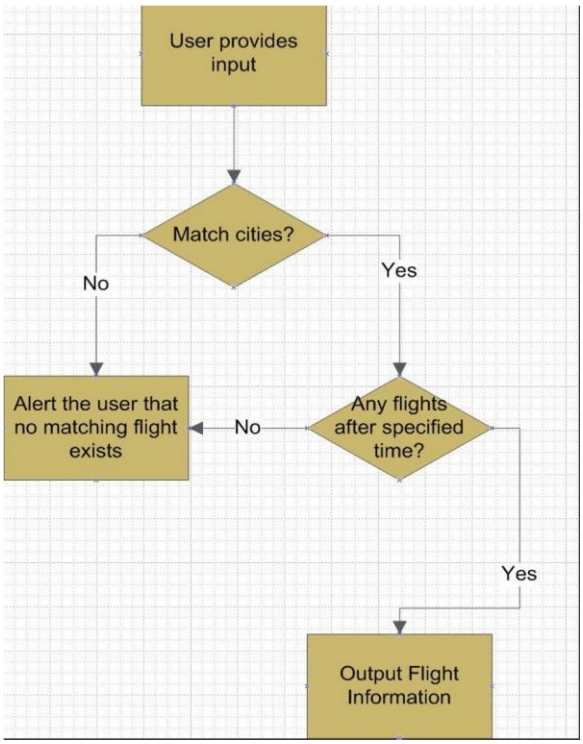
14.Enrollment

Description and Priority:

This function allows unregistered user to enroll and to create a new account with the website. In order to create a new account, the user has to provide required information such as first name, last name, email address and password. Other optional. information, such as phone number, credit card information and mailing address, can be provided during the registration process. The system checks if all required data are provided and then will prompt the user to enter additional information, if required. After all required information is provided, the system auto-generates a unique frequent flyer number that the user must use as username for future authentications. The system shall auto-generate this number in less than five seconds.

15.Activity Diagram:





16.My Account

Description and Priority:

This section gives the user the power to view, save, edit or delete the information stored in his/her account. The user can check his/her accumulated points, look at the status of a flight that was booked, cancel a flight that was already booked (optional) and change his/her address, phone number, email or password. This feature is not available for non-registered user.

16.1 External Interface Requirements

User Interfaces

A Help link will appear on every screen that describes the function of each page to the user. The implementation should be written so that blind users can still interact with the system (using a screen reader.)

16.2 Communications Interfaces

The system must utilize the standard Hyper Text Transfer Protocol (HTTP) to ensure maximum inter-browser compatibility. The client accesses the system through a web browser.

16.3 Other Nonfunctional Requirements

Performance Requirements

- The Airline Website shall have capabilities to accept 500 connections. For each session, system shall guarantee the connection time 5 minutes from last input, after which the connection will be deemed expired. A close operation will be performed when expired. This design is to satisfy each user's usability and connection quality.
- The system shall send out verification request immediately (within 100ms) after the it receives a user submitted form.
- The system shall update all flight status information every 5 minutes.

Security Requirements

- Passwords must be a minimum of eight characters and must contain one to seven digits.
- Email addresses should be verified before the system grants user access. This verification shall be exercised by sending the prospective user a confirmation email after enrollment. This email must contain information specific to completing the enrollment process.

16.4 External Interface Requirements

User Interfaces

A *Help* link will appear on every screen that describes the function of each page to the user. The implementation should be written so that blind users can still interact with the system (using a screen reader.)

Communications Interfaces

The system must utilize the standard Hyper Text Transfer Protocol (HTTP) to ensure maximum inter-browser compatibility. The client accesses the system through a web browser.

16.5 Other Nonfunctional Requirements

Performance Requirements

- The Airline Website shall have capabilities to accept 500 connections. For each session, system shall guarantee the connection time 5 minutes from last input, after which the connection will be deemed expired. A close operation will be performed when expired. This design is to satisfy each user's usability and connection quality.

- The system shall send out verification request immediately (within 100ms) after the it receives a user submitted form.

- The system shall update all flight status information every 5 minutes.

Security Requirements

- Passwords must be a minimum of eight characters and must contain one to seven digits.

- Email addresses should be verified before the system grants user access. This verification shall be exercised by sending the prospective user a confirmation email after enrollment. This email must contain information specific to completing the enrollment process.

17.Appendix:

The airline flight booking system outlined in this software requirements specification will provide a comprehensive and user-friendly platform for customers to search for flights, select their desired itinerary, make a reservation, and manage their bookings. The system will meet all applicable security and

performance requirements, and it will be designed to be scalable, maintainable, and extensible.

18. Software Development Methodologies

Ticket exchange platform software methodology is the approach taken to develop and maintain a software platform that allows users to buy and sell tickets to events. This methodology should include considerations for the following:

Secuirity: The platform must be secure to protect users' personal and financial information. This includes implementing measures to prevent fraud and unauthorized access to accounts.

Scalability: The platform must be able to handle a large volume of traffic and transactions. This is especially important for popular events where tickets sell out quickly.

User experience: The platform should be easy to use for both buyers and sellers. This includes having a clear and concise interface, as well as robust search and filtering capabilities.

Compliance: The platform must comply with all applicable laws and regulations. This includes things like data protection and consumer protection laws.

Here is a more detailed overview of the key steps involved in developing a ticket exchange platform:

1. **Requirements gathering**: The first step is to gather requirements from the users of the platform. This includes understanding what features they need and how they want the platform to work.
2. **System design:** Once the requirements have been gathered, the system can be designed. This includes defining the architecture of the platform, as well as the features and functionality that will be implemented.
3. **Development:** The next step is to develop the platform. This includes writing the code, testing the platform, and fixing any bugs.

4. **Deployment:** Once the platform is developed, it can be deployed to production. This includes making it available to users and configuring it to work with the existing infrastructure.
5. **Maintenance and support:** Once the platform is deployed, it needs to be maintained and supported. This includes fixing any bugs that are discovered, as well as adding new features and functionality as needed.

Here are some additional considerations for ticket exchange platform software methodology:

- **Fraud prevention**: Ticket exchange platforms are a prime target for fraudsters. It is important to implement measures to prevent fraud, such as verifying the identity of users and detecting counterfeit tickets.
- **Dispute resolution**: Ticket exchange platforms often have to deal with disputes between buyers and sellers. It is important to have a fair and efficient dispute resolution process in place.
- **Customer support**: Ticket exchange platforms should provide excellent customer support to their users. This includes being responsive to inquiries and resolving issues quickly and efficiently.

19.Testing

19.1 Proposed Testing Approach Compared to Others

Various testing approaches can be employed to evaluate software quality and ensure it meets the specified requirements. The choice of approach depends on the specific context, project goals, and available resources. Let's compare some common approaches:

Black-Box Testing vs. White-Box Testing:

- **Black-Box Testing:** This approach focuses on the external behavior of the software without considering its internal structure or implementation. Testers design test cases based on the software's specifications and requirements without knowledge of its code.
- **White-Box Testing:** This approach involves testing the software's internal logic, structure, and code. Testers design test cases based on the software's code and implementation to ensure it functions as intended.

Manual Testing vs. Automated Testing:

- **Manual Testing:** This approach involves testers manually executing test cases, observing the software's behavior, and recording any defects or deviations from expected behavior.
- **Automated Testing:** This approach utilizes tools and scripts to automate the execution of test cases, reducing manual effort and increasing the speed and consistency of testing.

Static Testing vs. Dynamic Testing:

- **Static Testing:** This approach analyzes the software's source code without executing it to identify potential defects, such as syntax errors, coding standards violations, and logical inconsistencies.
- **Dynamic Testing:** This approach involves executing the software to identify defects that manifest during runtime, such as functional errors, performance issues, and compatibility problems.

19.2 Discuss with Testing Levels

Software testing is typically conducted at different levels of abstraction, from individual components to the entire integrated system. The level of testing determines the scope and focus of the testing activities.

Unit Testing: This level focuses on testing individual units or components of the software, typically functions or classes, in isolation. It ensures that each unit behaves as expected according to its specifications.

Integration Testing: This level focuses on testing how different units or components interact with each other. It verifies that the interfaces between components work correctly and that data is exchanged seamlessly.

System Testing: This level focuses on testing the entire integrated system as a whole. It ensures that the system meets its overall functional and non-functional requirements, such as performance, security, and usability.

Acceptance Testing: This level is typically performed by users or stakeholders to validate that the system meets their acceptance criteria and is ready for deployment or release.

19.3 Testing Types, Techniques, and Tactics

Testing Types:

- **Functional Testing:** Verifies that the software functions as intended and meets the specified requirements.
- **Non-Functional Testing:** Evaluates non-functional aspects of the software, such as performance, security, usability, and reliability.

Testing Techniques:

- **Black-Box Testing Techniques**: Include test cases based on specifications, equivalence partitioning, boundary value analysis, and exploratory testing.
- **White-Box Testing Techniques:** Include code coverage analysis, control flow testing, data flow testing, and mutation testing.

Testing Tactics:

- **Top-Down Testing:** Starts from the high-level system and gradually moves down to individual components.
- **Bottom-Up Testing**: Starts from individual components and gradually builds up to the entire system.

19.4 Proposed Testing Process

The software testing process typically involves the following phases:

Planning: Defines the testing scope, objectives, resources, and schedule.

Test Case Design: Creates detailed test cases that specify inputs, expected outputs, and procedures.

Test Environment Setup: Sets up the hardware, software, and data necessary for testing.

Test Execution: Executes test cases and records results.

Defect Reporting and Management: Identifies, reports, and tracks defects until they are resolved.

Test Closure: Evaluates the overall effectiveness of testing and determines when to conclude.

19.5 Measurement in Software Testing

Hierarchy of Testing Difficulty:

A qualitative assessment of the difficulty of testing different software aspects:

- **Low Difficulty:** Testing basic functionality, data handling, and user interface interactions.
- **Medium Difficulty:** Testing complex interactions, error handling, and performance under load.
- **High Difficulty:** Testing security features, concurrency issues, and distributed systems.

Metrics (Test Plan, Test Case):

Quantitative measures to assess testing effectiveness:

- **Test Plan Coverage:** Percentage of requirements covered by test cases.
- **Test Case Execution Rate:** Percentage of test cases executed successfully.
- **Defect Detection Rate:** Percentage of defects detected during testing.

20.Technical Requirements

20.1 Front-end Languages:

HTML (HyperText Markup Language): HTML forms the foundation of web pages, defining the structure and content of the site. It provides the framework for displaying text, images, and other elements.

CSS (Cascading Style Sheets): CSS controls the presentation and styling of the website, determining the layout, colors, fonts, and overall visual appeal. It transforms the bare-bones HTML structure into an engaging and user-friendly interface.

JavaScript: JavaScript adds interactivity and dynamism to the website, enabling user interactions, animations, and complex functionalities. It enhances

the user experience by making the site responsive and responsive.

20.2 Back-end Languages:

Python: Python is a popular choice for eCommerce back-end development due to its versatility, scalability, and ease of use. Its extensive libraries and frameworks, such as Django and Flask, make it efficient for building robust and secure web applications.

PHP: PHP is another widely used back-end language, particularly for eCommerce platforms like WooCommerce and Magento. Its simplicity and integration with MySQL databases make it a suitable option for managing product catalogs, user accounts, and order processing.

Java: Java is a powerful and enterprise-level language, often used for large-scale eCommerce systems. Its stability, performance, and security features make it ideal for handling high traffic and complex transactions.

20.3 Database:

MySQL: MySQL is a widely used open-source relational database management system (RDBMS), renowned for its ease of use, performance, and reliability. It's an excellent choice for storing and managing eCommerce data, including product information, customer details, and order history.

PostgreSQL: PostgreSQL is another open-source RDBMS known for its advanced features, scalability, and data integrity. It offers robust transaction management and supports a variety of data types, making it suitable for handling complex eCommerce requirements.

MongoDB: MongoDB is a NoSQL database that stores data in flexible JSON-like documents, making it ideal for managing unstructured and frequently changing data. It's
popular choice for eCommerce platforms that deal with large volumes of product data, user reviews, and customer preferences.

In summary, the choice of front-end and back-end languages, as well as the database, depends on the specific requirements and complexity of the eCommerce project. Factors to consider include the size of the project, the level of customization, the expected traffic volume, and the security needs.

20.4 Security:

Security is paramount for an online shopping system, as it handles sensitive customer information and financial transactions. Essential security measures include:

User authentication and authorization: Implement secure mechanisms for user registration, login, and access control to prevent unauthorized access to user accounts and sensitive data.

Data encryption: Encrypt sensitive data, such as credit card information and passwords, both at rest (stored in the database) and in transit (during transmission over the network).

Input validation: Sanitize and validate user input to prevent malicious code injection (SQL injection, cross-site scripting) and protect against data tampering.

Regular security updates: Regularly update software and libraries to patch vulnerabilities and mitigate known security risks.

20.5 Model: MVC V5:

MVC (Model-View-Controller) is a software design pattern that separates the application's concerns into three distinct components:

Model: The model represents the data layer, responsible for managing and storing data in the database.

View: The view handles the presentation layer, responsible for rendering the user interface and displaying data from the model.

Controller: The controller acts as the intermediary between the model and the view, handling user interactions, updating the model, and triggering the view to reflect changes.

MVC V5 is the latest version of the MVC pattern, providing improvements in modularity, maintainability, and testability.

21.Payment Terms & condition

15% payment will be accepted for the Project proposal and design Submission.

45% payment will be accepted for the Application Development

70% payment will be accepted after application review and Testing

100% payment will be accepted after handover the fully completed Application

22.Responsibility

The entire Application has been done by MMK Mahin and all the responsibility including terms and condition will goes to him.

23.Contact Us

You can get in touch with us in any of the below ways:

Golam Rabbany
By Phone: +8801787774996

By Email
grabbany1234@gmail.com

23.Agreement Signed By:

……	……..	……
Client Signature Ashif Hasan Managing Director	Order Provider Signature Rabbi hassan Officer SoftTech	Authority Signature Golam Rabbany Managing Director (MD) SoftTech