

# **INTELLISCORE - Ai Based Credit Scoring System**

This BS Project report is submitted to the Department of Computer Science as partial fulfillment of Bachelor of Science in Computer Science degree

*<https://github.com/Fazulsden/Fyp-SD.git>*

by

**Shahmeer Khan, Fazul Al Rehman. Noshwan Sheikh, Abdur Rehman**

Advised by

**Miss Tasbiha Fatima**

Lecturer

Department of Computer Science

School of Mathematics and Computer Science (SMCS)

Institute of Business Administration (IBA) Karachi

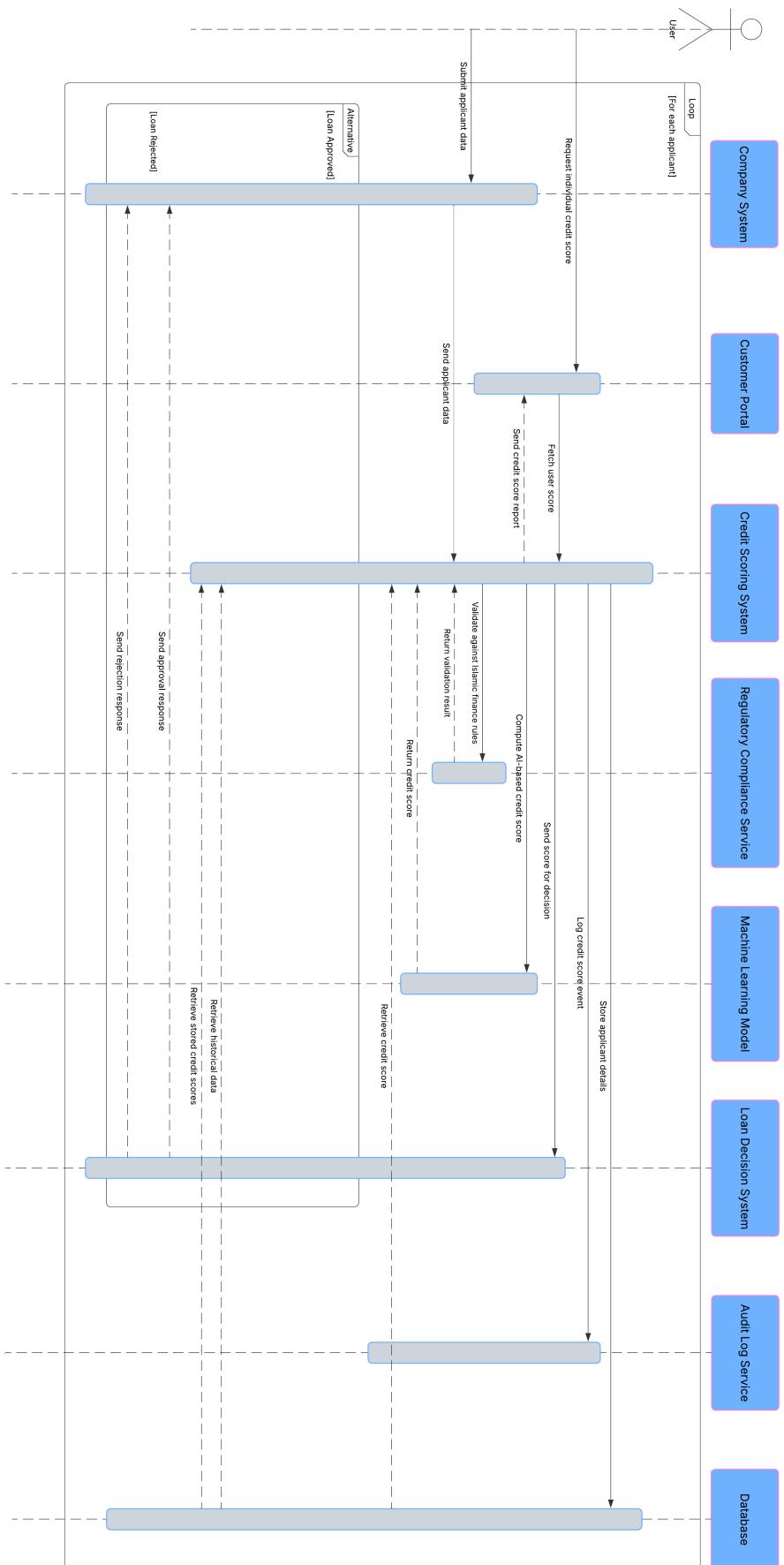
# Contents

<b>1</b>	<b>Interaction Diagrams</b>	<b>1</b>
<b>2</b>	<b>Class Diagram and Interface Specification</b>	<b>3</b>
<b>3</b>	<b>System Architecture and System Design</b>	<b>4</b>
3.1	Architectural Style . . . . .	4
3.2	Identifying Subsystems . . . . .	5
3.3	Mapping Subsystems to Hardware . . . . .	5
3.4	Persistent Data Storage . . . . .	6
3.5	Network Protocol . . . . .	7
3.6	Global Control Flow . . . . .	7
3.7	Hardware/Sofware Requirements . . . . .	8
<b>4</b>	<b>Algorithms and Data Structures</b>	<b>9</b>
<b>5</b>	<b>User Interface Design and Implementation</b>	<b>12</b>
5.1	UI diagrams . . . . .	12
5.2	HCI Principles Followed . . . . .	15
<b>6</b>	<b>Design of Tests</b>	<b>17</b>

# **Chapter 1**

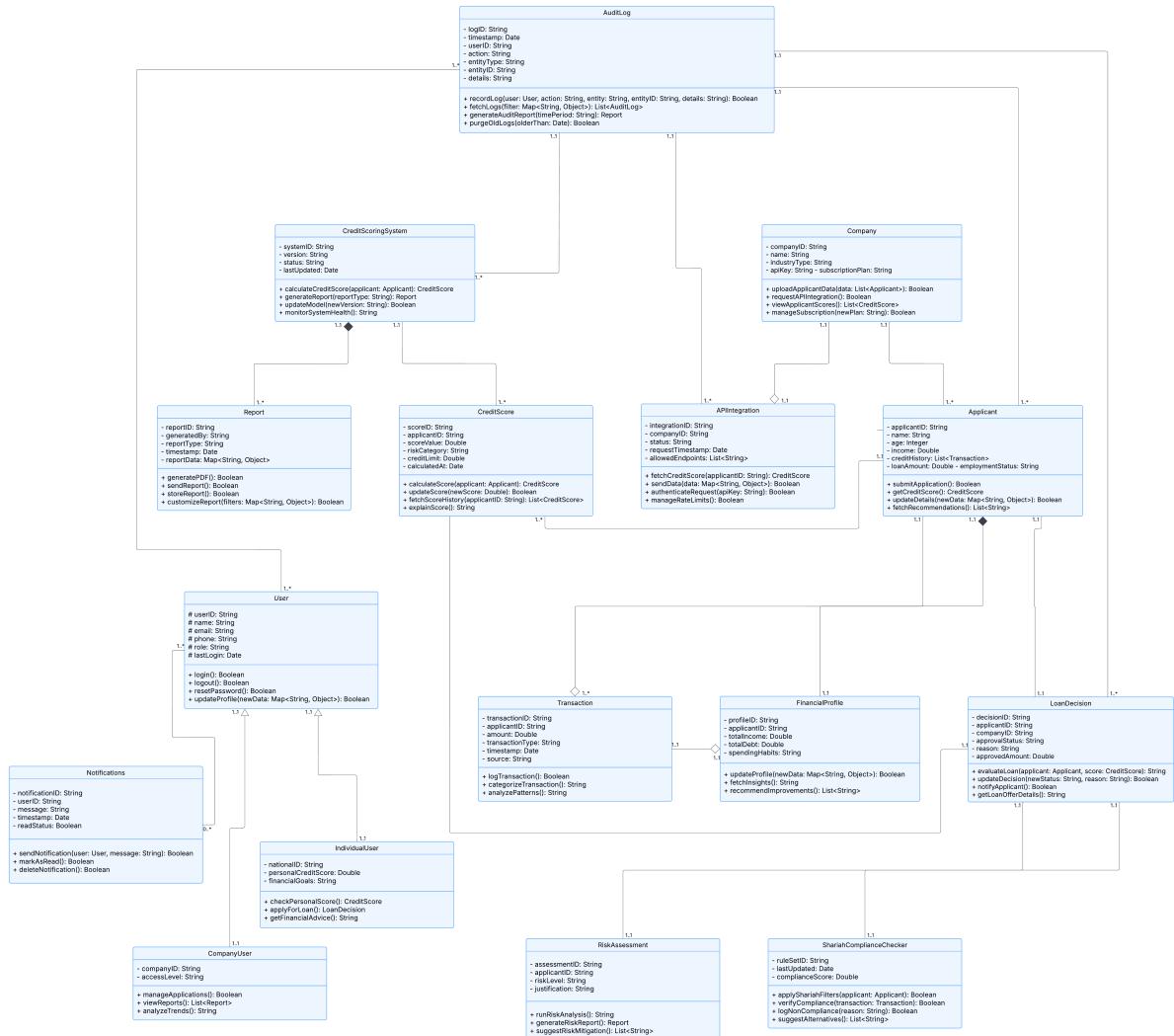
## **Interaction Diagrams**

As the diagram is fairly large it has been displayed on the following page horizontally i.e. top to bottom being left to right to view the full sequence.



# Chapter 2

## Class Diagram and Interface Specification



# Chapter 3

## System Architecture and System Design

### 3.1 Architectural Style

We have Selected Client-Server Architecture because it offers key advantages for an AI-based credit scoring system. It centralizes business logic, algorithms, and data storage, simplifying updates and ensuring consistency also system is scalable, The server handles resource-intensive tasks like running AI algorithms, improving efficiency, while the client focuses on the user interface. This architecture ensures security, efficiency, and scalability for credit scoring.

Layers in Client-Server Architecture

#### 1. Presentation Layer (Client-Side):

This layer is responsible for the user interface where users input financial information and view results like credit scores. It sends requests to the server and displays the processed data.

#### 2. Application Layer (Server-Side):

The core of the system, this layer processes requests, runs AI algorithms, and handles the credit scoring logic. It ensures that the application responds based on the input data and integrates business rules.

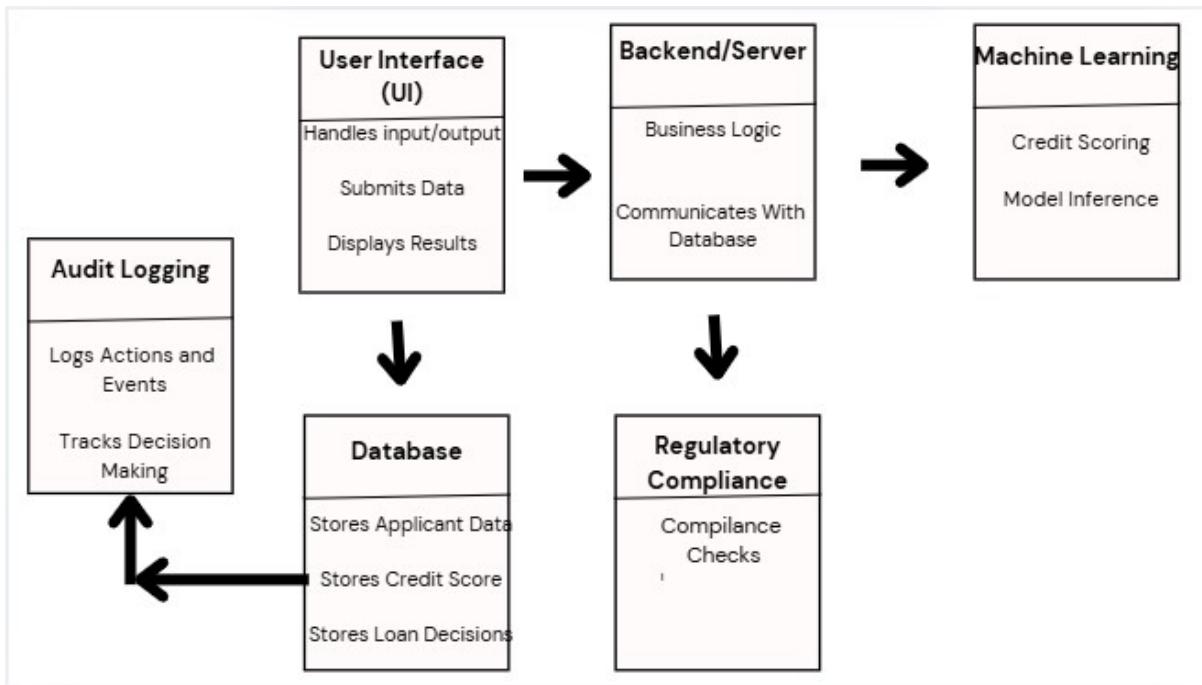
#### 3. Data Layer (Server-Side):

This layer manages the storage and retrieval of data required for credit scoring. It includes database systems that store user financial history, transaction data, and scoring records, which are then accessed by the application layer.

#### 4. Security Layer (Cross-Cutting Concern):

Security is integrated across all layers, ensuring encrypted communication and secure access to sensitive financial data. It involves firewalls, encryption protocols, and authentication systems to protect user information.

## 3.2 Identifying Subsystems



## 3.3 Mapping Subsystems to Hardware

- Frontend Layer (Client Access): Client devices (PCs, Phones) will interact with the system. The frontend runs on the client-side, accessing the backend and ML models hosted in the cloud via the internet
- Backend Layer (Cloud Infrastructure): Amazon Aws (or Vercel) that would host backend. The backend would run on cloud servers, managing user requests, authentication, and routing to the database and machine learning model.
- Database Layer (MySQL on Cloud): Cloud-managed database instances. If budget is lowered MongoDB Might be used (Cloud). The MySQL database runs on cloud servers storing and retrieving data for the application.
- Machine Learning Model Backend: Cloud GPUs/TPUs or specialized ML instances. Machine learning inference happens on cloud-based hardware optimized for computation, in the start it would be hosted on an free cloud (possibly collab) later if used properly sagemaker.

## 3.4 Persistent Data Storage

### 1. Applicants Table

Column Name	Data Type	Description
applicant_id (PK)	VARCHAR	Unique identifier for the applicant.
name	VARCHAR	Full name of the applicant.
age	INT	Age of the applicant.
income	DECIMAL(10, 2)	Applicant's monthly/annual income.
employment_status	VARCHAR	Employment status (e.g., employed, self-employed).
credit_history	VARCHAR	Credit history (e.g., good, poor).
loan_amount_requested	DECIMAL(10, 2)	Requested loan amount.
loan_type	VARCHAR	Type of loan requested (e.g., personal, mortgage).
application_date	TIMESTAMP	Date and time when the application was submitted.

### 2. Credit Scores Table

Column Name	Data Type	Description
score_id (PK)	SERIAL	Unique identifier for the credit score entry.
applicant_id (FK)	VARCHAR	Foreign key referencing the applicant's applicant_id.
credit_score	INT	The AI-generated credit score.
score_date	TIMESTAMP	Date and time when the credit score was calculated.
model_used	VARCHAR	The AI model used to calculate the credit score (e.g., Random Forest, SVM).
score_validated	BOOLEAN	Flag to indicate if the score was validated (True/False).

### 3. Regulatory Compliance Table

Column Name	Data Type	Description
compliance_id (PK)	SERIAL	Unique identifier for the compliance check.
applicant_id (FK)	VARCHAR	Foreign key referencing the applicant's applicant_id.
compliance_result	BOOLEAN	Whether the applicant complies with Islamic finance principles (True/False).
check_date	TIMESTAMP	Date when the compliance check was performed.

#### 4. Loan Decisions Table

Column Name	Data Type	Description
decision_id (PK)	SERIAL	Unique identifier for the loan decision entry.
applicant_id (FK)	VARCHAR	Foreign key referencing the applicant's applicant_id.
loan_decision	VARCHAR	Decision made: Approved or Rejected.
decision_date	TIMESTAMP	Date and time when the loan decision was made.
decision_reason	TEXT	Reason for the loan decision (e.g., credit score insufficient, compliance issue).

#### 5. Audit Log Table

Column Name	Data Type	Description
log_id (PK)	SERIAL	Unique identifier for the log entry.
action_type	VARCHAR	Type of action performed (e.g., submit, validate, score).
timestamp	TIMESTAMP	Date and time when the action occurred.
user_id	VARCHAR	ID of the user performing the action.
details	TEXT	Detailed description of the action (e.g., "Data submitted for credit scoring").

### 3.5 Network Protocol

#### HTTPS

Secure communication between the frontend (React) and backend (Django) for submitting applicant data, requesting credit scores, and receiving loan decisions.

#### RESTful API

Backend will expose RESTful APIs for operations like submitting data, fetching credit scores, and handling loan decisions, using JSON for data exchange.

#### SQL over TCP/IP

Backend communicates with the database (PostgreSQL/MySQL) using SQL queries over a secure TCP/IP connection

#### SSL/TLS Encryption

Ensures secure transmission of sensitive data over the network, maintaining confidentiality and data integrity.

### 3.6 Global Control Flow

The system will follow an event-driven execution model, triggered by user actions like submitting data or requesting credit scores.

**Time dependency:**

Timers will be used for delays like waiting for regulatory compliance checks or ML model computation

**Concurrency:**

Multiple threads will handle concurrent requests, ensuring smooth processing of multiple applicants. Asynchronous task handling will be used for operations like score calculation and loan decision.

## 3.7 Hardware/Sofware Requirements

Platform	Component	Minimum Requirements	Recommended Requirements
Android	Hardware	2 GB RAM, ARM-based processor (1.8 GHz octa-core), 2 GB free storage	4 GB RAM, ARM-based processor (2.0 GHz or higher), 2 GB free storage
Android	Software	Android 5.0 (Lollipop) or higher, HTTPS for secure communication	Android 10.0 or higher, HTTPS for secure communication
Windows PC	Hardware	4 GB RAM, Intel Core i3 (or equivalent AMD), 10 GB free storage	8 GB RAM, Intel Core i5 (or equivalent AMD), 10 GB free storage
Windows PC	Software	Windows 10 (64-bit), HTTPS for secure communication	Windows 10 (64-bit), HTTPS for secure communication

# Chapter 4

## Algorithms and Data Structures

### 1. Neural Networks for Credit Scoring

Neural networks have been chosen as the primary machine learning algorithm for our credit scoring system because they can recognize complex relationships in financial data. Credit scoring involves analyzing large amounts of structured financial data, such as transaction histories, income levels, and credit histories. Neural networks are particularly useful because they can detect hidden patterns and interactions within these features. The model includes multiple layers, each processing data to improve accuracy.

### 2. Previous Approaches and Their Limitations

During our testing, we also tried simpler methods like k-Nearest Neighbors (k-NN) and decision trees. However, they did not perform as well as neural networks for handling financial data.

### 3. k-Nearest Neighbors (k-NN)

k-NN works by finding data points that are closest to a given input and using them to make predictions. The underlying data structures used in k-NN are arrays, which help organize and search data efficiently. However, financial data is often high-dimensional, meaning it has many features. This causes k-NN to perform poorly because it struggles with too many variables, making distance-based predictions unreliable. Additionally, k-NN does not handle categorical data well, which is common in financial records.

### 4. Decision Trees

Decision trees work by making decisions based on yes/no questions about different financial attributes. The data structure used in decision trees is a tree, where each decision is a node that splits into possible outcomes. While decision trees are easy to understand, they often overfit, meaning they memorize the training data instead of learning useful patterns. Since financial data is affected by many external factors like market conditions, decision trees do not generalize well.

## **Data Preprocessing and Simple Data Structures Used**

Before training the neural network, we apply preprocessing steps to clean and prepare the data. These steps use fundamental data structures such as:

- Arrays: Used to store and process lists of numerical values, such as customer income and loan amounts.
- Queues: Used to handle sequential processing tasks, such as reading and processing financial records in batches.

Steps in preprocessing include:

### **1. Handling Missing Data:**

If financial records have missing values, we either fill them with estimated values or remove incomplete entries. Arrays store the cleaned data.

### **2. Encoding Categorical Data:**

Since machine learning works with numbers, we convert categories like "Good" and "Bad" credit history into numerical values using hash tables.

### **3. Feature Scaling:**

Some financial attributes, like income, have much larger values than others, like interest rates. To ensure fair treatment of all attributes, we normalize them using mathematical transformations stored in arrays.

### **4. Outlier Detection:**

We remove data points that seem unusual or incorrect, such as an income value of PKR 100,000,000, which could be an error. This is done using sorting and searching techniques within arrays.

## **Data Structures in Neural Networks**

Neural networks rely on advanced mathematical structures, but at their core, they use basic data structures such as:

- Tensors: These are multi-dimensional arrays that store input features, weights, and activations.
- Matrices: Used extensively for performing calculations during training, such as adjusting model weights.

- Queues and Stacks: Used in training processes like backpropagation, where data flows through multiple layers in an organized way.

## **Algorithms and Data Structures in a Deployed Web Application**

When the system is live, it will consist of a frontend (FE), backend (BE), database (DB), and a neural network server running separately. Since we are using industry-standard libraries, many data structures and algorithms will be handled internally. However, certain optimizations and custom implementations will still be necessary.

The frontend will handle user interactions and display results. Arrays and objects will store UI components and input data, while queues will help with asynchronous data fetching. The backend will manage business logic and database interactions, using hash tables for caching, trees for indexing, and queues for processing tasks. The database will use Binary Trees and hash tables for indexing and efficient queries, with SQL joins to fetch and merge financial data dynamically. The neural network server will handle credit score predictions using tensors, matrix operations, and dynamic programming techniques like backpropagation.

## **Custom Implementations**

While libraries handle most operations, some components will require manual implementation. Custom data pipelines will be necessary to clean and format data before passing it to the model. Custom caching mechanisms may be required to speed up response times for frequently accessed credit scores.

## **Why Neural Networks Work Best**

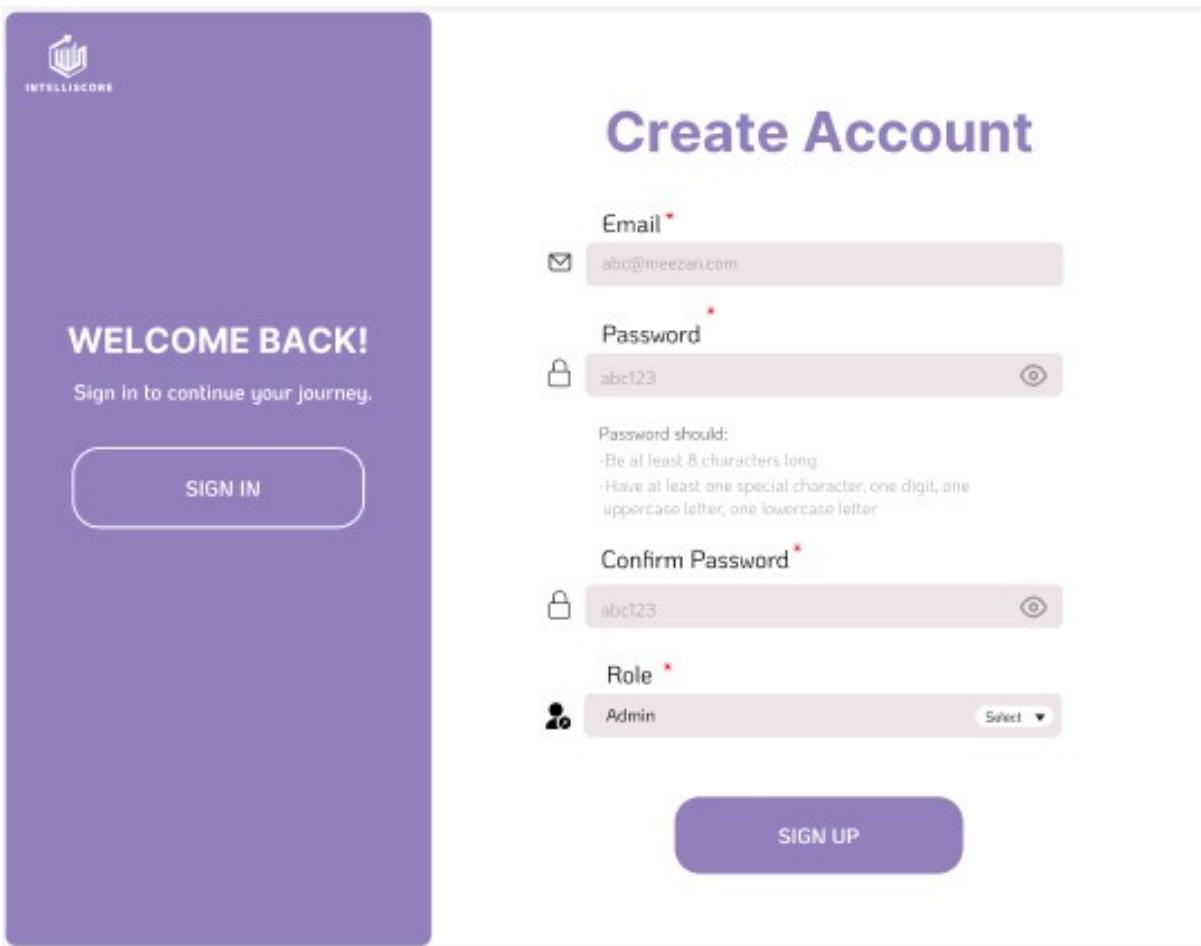
Neural networks perform better than k-NN and decision trees because they adapt to complex financial data patterns. Unlike decision trees, which follow fixed decision paths, neural networks adjust dynamically to find the best patterns in the data. Unlike k-NN, which struggles with many features, neural networks process financial attributes efficiently using matrix operations.

By leveraging these data structures and algorithms, we can create an accurate credit scoring model that helps financial institutions make informed loan decisions, reducing risks and improving approval processes. A large part of these implementations will be handled by the frameworks and libraries we use, so we will not need to build everything from scratch using primitive data structures and algorithms.

# Chapter 5

## User Interface Design and Implementation

### 5.1 UI diagrams



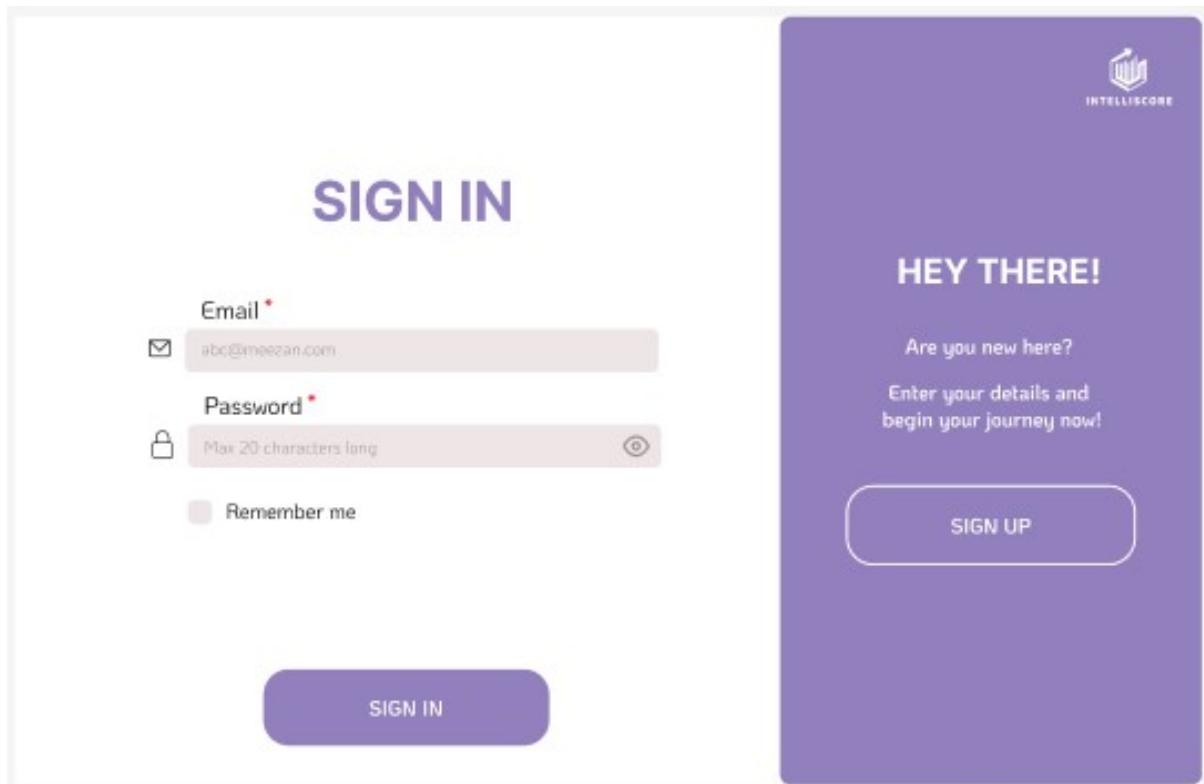
The screenshot shows the 'Create Account' page of an application. On the left, there is a purple sidebar with the 'INTELLISCORE' logo at the top. Below it, the text 'WELCOME BACK!' is displayed in large white capital letters, followed by the subtext 'Sign in to continue your journey.' and a 'SIGN IN' button.

The main content area has a light gray background. At the top, the title 'Create Account' is centered in a large, bold, blue font. Below the title are four input fields:

- Email \***: An input field containing the value 'abc@meezan.com' with a small mail icon to its left.
- Password \***: An input field containing the value 'abc123' with a small lock icon to its left. To the right of the input field is a circular 'eye' icon for password visibility.
- Confirm Password \***: An input field containing the value 'abc123' with a small lock icon to its left. To the right of the input field is a circular 'eye' icon for password visibility.
- Role \***: A dropdown menu currently set to 'Admin' with a person icon to its left. To the right of the dropdown is a 'Select' button with a downward arrow.

Below these fields, there is a note: 'Password should:  
-Be at least 8 characters long  
-Have at least one special character, one digit, one uppercase letter, one lowercase letter'. At the bottom right of the main area is a large, rounded purple 'SIGN UP' button.

(a) Signup Page



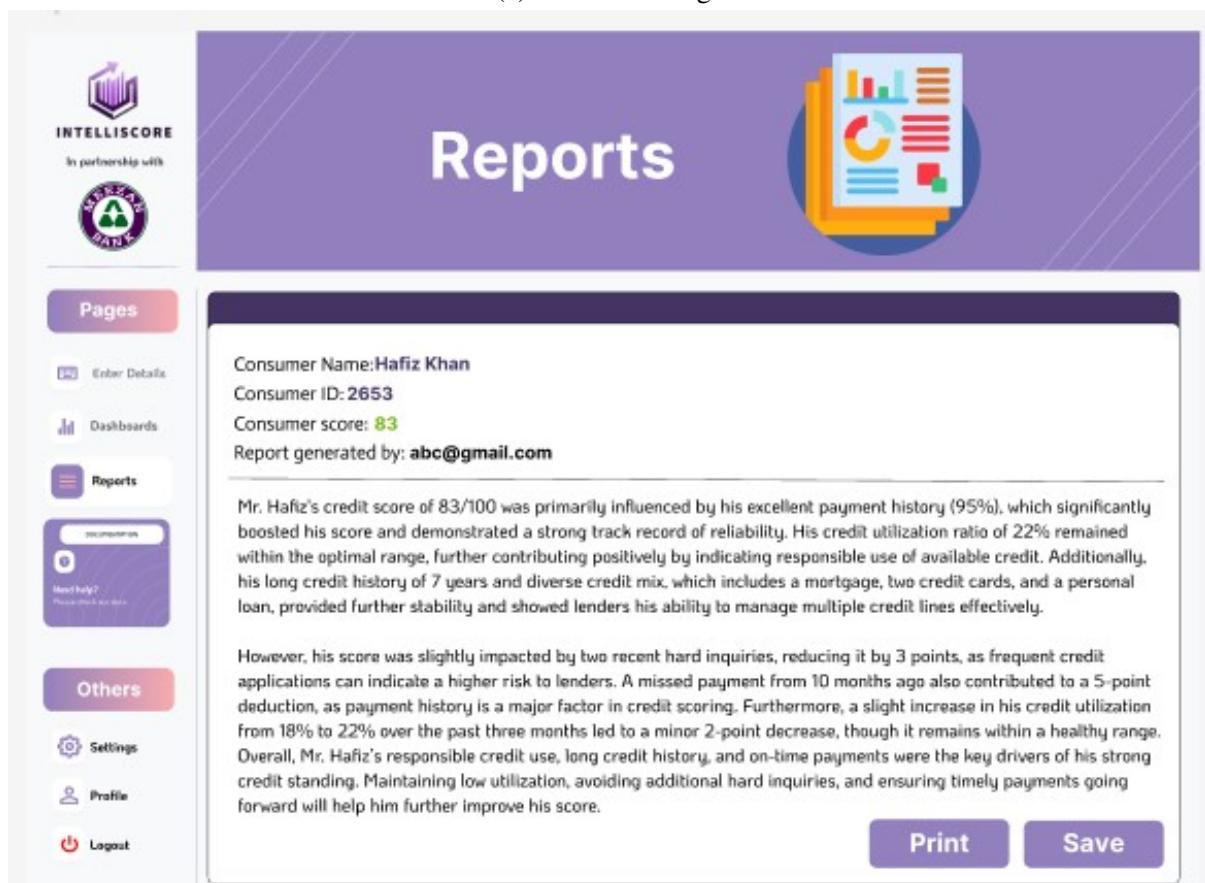
(a) Signin Page

The Data Entry page has a sidebar with 'INTELLISCORE' and 'In partnership with MEEZAN BANK' logos, and navigation links for 'Pages', 'Enter Details', 'Dashboards', 'Reports', and 'Documentation' (with a note about checking for updates). The main content area features a large 'Data Entry' title with a keyboard and document icon. A 'Rules to follow' section lists instructions: 'Kindly fill this form with accurate data as checks will be performed for everything submitted', 'Failure to comply and attempts at fraud could result in serious consequences', 'Questions marked with a \* must be filled, questions without this mark are optional but will aid in a more accurate analysis', and 'Questions that need documents submitted will have an option to upload data'. Below this, a note says 'Once the form is filled, click generate to get your credit score as you will be leaded to the dashboards screen'. The page contains two input fields for 'Q1 What is your monthly income?' and 'Q1 What is your occupation?', both with 'Click here to type' placeholder text. A large purple 'Generate' button is located at the bottom right.

(b) Data Entry Page



(a) Dashboard Page



(b) Report Page

## **5.2 HCI Principles Followed**

### **1. Consistency (Aesthetic and Minimalist Design)**

Applied in: All Screens

- The design follows a consistent color scheme (purple and white), typography, and button styles across all screens, ensuring a familiar user experience.
- Icons and layout are similar across pages, which helps users recognize patterns and reduces cognitive load.

### **2. Visibility of System Status**

Applied in: Dashboard Screen

- The dashboard provides real-time feedback on the credit score, financial statistics, and trends, ensuring the user is always aware of their financial status.
- Visual indicators like graphs and score display help users interpret their financial standing and scoring metrics at a glance.

### **3. User Control and Freedom**

Applied in: Reports and Data Entry Screens

- The "Print" and "Save" buttons in the Reports screen give users control over how they handle their reports.
- The Dashboards and reports screens give the users control over what format they want the data to be visualized in catering for both detailed and general analysis.

### **4. Error Prevention**

Applied in: Sign-up and Sign-in Screens

- Password requirements are explicitly mentioned to prevent users from entering weak passwords.
- Email validation is present to avoid incorrect email entries.
- Mandatory fields are marked with asterisks (\*) to ensure users fill in necessary details before proceeding.

### **5. Recognition Rather than Recall**

Applied in: Dashboard and Reports Screens

- The dashboard includes financial metrics displayed in an intuitive way, such as labeled graphs and statistics, helping users quickly understand their credit performance without needing to recall past data.

- The Reports screen presents an explanation of the credit score, reducing the need for users to remember scoring criteria.

## **6. Flexibility and Efficiency of Use**

Applied in: Dashboard and Reports Screens

- Users can generate reports, save them, or print them for later use, adding efficiency.
- The dashboard includes multiple financial insights and graphs that accommodate both inexperienced users through visual elements and expert users through numerical details as well as the reports page for more detailed insights.

## **7. Match Between System and the Real World**

Applied in: All Screens

- Terminologies used in the forms (e.g., “Monthly Income” and “Occupation”) are standard financial terms that users can easily understand.
- The dashboard uses familiar data visualization techniques like line graphs and bar charts, which resemble real-world financial reports.
- Throughout the screens various metaphors are used to signal real life comparisons aiding in easier understanding and familiarity i.e. an email icon for the email field, a power button for logout option etc

## **8. Help Users Recognize, Diagnose, and Recover from Errors**

Applied in: Sign-up and Data Entry Screens

- Error handling is evident in the password creation process, where users are informed about requirements.
- Email validation is present to avoid incorrect email entries.
- The form validation in Data Entry ensures users enter valid details before submitting.

In comparison to our original low level wireframes where functionality was preferred over ease of use and aesthetics, the updated ui system incorporates all the commercially used HCI standards as well as integrating the colour scheme of Meezan Bank to follow consistency as the app is meant for the bank's use.

# Chapter 6

## Design of Tests

The following chapter will discuss the test cases that we will validate our final software on, for better understanding we have added a status column as-well that will be filled after testing is complete on the final product.

1	Test Case Type	Description	Test Step	Expected Result	Status (Pass/Fail)
2	Backend & Database Integrity	Check database connection	Connect to DB	Successful connection	
3	Backend & Database Integrity	Verify data consistency	Query for duplicate records	No duplicates found	
4	Backend & Database Integrity	Check API response time	Measure API latency	Response time < 500ms	
5	Backend & Database Integrity	Validate user authentication	Login with valid credentials	Access granted	
6	Backend & Database Integrity	Verify transaction rollback	Trigger rollback scenario	Database remains consistent	
7	Web Application Functionality & Security	Test login functionality	Enter valid credentials	Successful login	
8	Web Application Functionality & Security	Check SQL injection prevention	Try SQL injection in login	Attack blocked	
9	Web Application Functionality & Security	Verify password hashing	Inspect stored passwords	Passwords are hashed	
10	Web Application Functionality & Security	Ensure role-based access control	Access admin panel as user	Access denied	
11	Web Application Functionality & Security	Check session timeout	Remain idle for set time	Auto logout	
12	Data Preprocessing and Cleaning	Handle missing values	Run data cleaning script	Missing values handled	
13	Data Preprocessing and Cleaning	Normalize numeric features	Apply feature scaling	Data scaled correctly	
14	Data Preprocessing and Cleaning	Detect and remove outliers	Run anomaly detection	Outliers removed	
15	Data Preprocessing and Cleaning	Convert categorical variables	Apply encoding techniques	Categories converted	
16	Data Preprocessing and Cleaning	Ensure data format consistency	Check column data types	Consistent formats	
17	Machine Learning Model Validation	Verify model accuracy	Run test dataset	Accuracy > 85%	
18	Machine Learning Model Validation	Test model under real-world conditions	Feed live data	Correct classifications	
19	Machine Learning Model Validation	Ensure model does not overfit	Evaluate training vs. test accuracy	No significant variance	
20	Machine Learning Model Validation	Check model response time	Measure prediction latency	Response time < 1s	
21	Machine Learning Model Validation	Test adversarial robustness	Introduce noisy inputs	Resilient predictions	

(a) Test Cases