הקדמה

המערכת שבחרנו הינה מערכת לניהול חברה ליצור רכבים. המערכת כולל את המפעלים שמייצרים את הרכבים וגם את תחנות המכירה. המערכת תוכל לנהל מידע על מודלי הרכבים עובדים מתקני הייצור. המטרה של המערכת היא לייעל ולקל על ניהול החברה.

**תיאור מבנה המערכת**

**חברה:** משמשת כישות העיקרית, מחזיקה מידע על שם החברה, מיקום המוקד הראשי, ורשימה של מפעליה, חנויותיה, ומודלי הרכבים שלה.

**מודל רכב:** מתאר מודלי רכב אישיים המיוצרים על ידי החברה, כולל נתונים כמו שם המודל, מפרטים, רכיבים זמינים, וזיהוי מפעל הייצור.

**עובד:** מייצג עובדי החברה, עם פרטים כמו זיהוי עובד, שם, תפקיד, והמפעל או החנות שבה הם מוקצים.

**מפעל:** מתאר את אתרי הייצור של החברה, כולל זיהוי מפעל, מיקום, קיבולת הייצור, ורשימה של מודלי הרכבים המיוצרים שם.

**רכיב רכב:** מפרט את הרכיבים המשמשים במודלי הרכב, כגון סוגי מנוע, הילוכים, ותכונות פנימיות, כולל זיהוי רכיב, שם, ומפרטים ומחיר

**חנות:** מייצג חנויות מזכירות אשר מוכרות את רכבי החברה, עם מידע על זיהוי חנות, מיקום, מלאי של מודלי הרכב, ונתוני מכירות.

**רשת ההפצה:** מתמקדת בלוגיסטיקת ובכננות של החברה, פרטים על זיהוי רשת, סוגי התחבורה, האזוריים, ומסלולי ההפצה.

**דחיסה של מבני נתונים לבינארי:**

נדחוס את כל העובדים לקובץ בינארי באופן הבא:

הנחנו שבשמות העובדים יהיו רק אותיות ולכן ניתן היה לשמור אותם כ 5 ביטים ושאורך השם אינו עולה על 256. עוד הנחנו שבחברה אין יוצר מ 65536 עובדים ולכן מספיקים 16 ביטים. בנוסף משכורת חודשית אינה עולה על מיליון שקל ומתחלקת באלף לכן השתמשנו בשש עשרה ביטים ושמרנו את מספר אחרי שלושת האפסים. לבסוף הנחנו שלחברה אין יותר מ256 מפעלים וחנויות. למחלקות ולאורכים לא הנחנו דבר כדי לאפשר חופש בבחירת השמות למפעלים וכו'.

יצוג עובד:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| תפקיד  1B\*כמות | שם מחלקה  אורך B | אורך מחלקה  4B | מספר מפעל  2B | משכורת  2B | מס עובד  2B | שם העובד  אורך השם \* 5  b | אורך השם  2B |

**פעולות משתמש:**

**הצגת ועריכת נתוני המפעל -** מאפשרת גישה לפרטים אודות כל מפעל תחת החברה.

**הוספת מודלים חדשים לחברה -** מאפשרת להוסיף מודל רכב חדש לקטלוג המודלים של החברה, כולל פרטים כמו שם המודל, מפרט טכני, ומחיר.

**הוספה של עובדים** -הוספת עובד למערכת ותיאום העובד למפעל או החנות הרלוונטיות.

**הוספה של מפעלים** - הוספה של מפעל ותיאום עם הרשת הפצה הרלוונטית..

**הוספה של חנויות** – הוספה של חנות ותיאום עם הרשת הפצה הרלוונטית.

הוספה של רכיבים לרכבים **– הוספה למפעל רכיב שהוא מיצר.**

הוספה של מודל רכב חדש **הוספת מודל חדש למפעל בהתאם לרכיבים הרלוונטים.**

**מיון ל עובדים לפי שם מס' עובד ופרטי העבודה**- יכולת למיין את החנויות, המפעלים או העובדים לפי קריטריונים שונים כגון מכירות, כמות עובדים או אלפביתית.

**חיפוש עובד -** חיפוש במערכת על מנת למצוא פרטים ספציפיים על עובדים

**פיטור עובד- אפשרות לפטר עובד מהחברה**

A close-up of a computer screen

Description automatically generated

מבנה המערכת:

char\* company\_name;

ShopFactoryNode\* head\_shops\_and\_factories // nested list

Worker\* CEO;

int num\_shops\_and\_fact; // Total number of shops and factories

Worker\*\* workers; // Array of workers

int num\_workers;

int max\_workers;

int id\_counter;

DistributionNetwork south, north; // Distribution networks

void init\_company(Company\* company, const char\* name);

void change\_CEO(Company\* company, Worker\* ceo);

void add\_shop\_factory\_node(Company\* company, void\* shop\_or\_factory, int is\_shop);

Worker\* add\_worker(Company\* company, const char\* name, const char\* position, double salary, const char\* department, int factory\_id);

ShopFactoryNode\* find\_factory\_shop\_by\_id(Company\* company, int factory\_id);

CarModel\* find\_model\_by\_name(Company\* company, const char\* model\_name);

int find\_worker\_by\_id(Company\* company, int worker\_id);

void free\_company(Company\* company);

void move\_worker(Company\* company, int worker\_id, int factory\_shop\_id);

void fire\_worker(Company\* company, int worker\_id);

void display\_workers\_company(Company\* company);

Struct CarModel:

Char\* model\_name[30]

Int model\_year

Char\* manufacturer

Double price

CarComponent\*\* components;

int num\_components

Int production\_volume

Functions:

Void update\_model\_name(CarModel\* model, Char\* new\_name);

Void update\_model\_year(CarModel\* model, Int new\_year);

Void update\_manufacturer(CarModel\* model, Char\* new\_manufacturer);

Void update\_price(CarModel\* model, Double new\_price);

Void add\_component(CarModel\* model, CarComponent\* component);

Void remove\_component(CarModel\* model, Int componentIndex);

Int calculate\_production\_cost(CarModel\* model);

Void display\_model\_info(CarModel\* model);

Struct Shop:

Int shop\_id;

Char\* location

Car\_Models\*\* inventory

Double sales

Worker\*\* workers

Functions:

Void update\_location(Shop\* shop, Char\* new\_location)

Void add\_to\_inventory(Shop\* shop, Car\_Models\* model)

Void remove\_from\_inventory(Shop\* shop, Int modelIndex)

Void update\_sales(Shop\* shop, Double sales\_amount)

Void display\_shop\_info(Shop\* shop)

Struct DistributionNetwork:

Int network\_id

Char\* description

Factory\*\* source\_factory

Shop\*\* destination\_shops

Int routes\_count;

Functions:

Void update\_description(DistributionNetwork\* network, Char\* new\_description);

Void add\_destination\_shop(DistributionNetwork\* network, Shop\* shop);

Void remove\_destination\_shop(DistributionNetwork\* network, Int shopIndex);

Void update\_routes\_count(DistributionNetwork\* network, Int new\_count);

Void display\_network\_info(DistributionNetwork\* network);

Struct Factory:

Int factory\_id

Char\* location

Int production\_capacity

Car\_Models\* produced\_models

CarComponent\* produced\_components

Worker\*\* workers

Functions:

Void update\_location(Factory\* factory, Char\* new\_location)

Void update\_production\_capacity(Factory\* factory, Int new\_capacity)

Void add\_produced\_model(Factory\* factory, Car\_Models\* model)

Void remove\_produced\_model(Factory\* factory, Int modelIndex)

Void display\_factory\_info(Factory\* factory)

Struct Worker:

Char\* worker\_name

Int worker\_id

Char\* position

Double salary

Char\* department

Int factory\_id

Functions:

Void update\_worker\_name(Worker\* worker, Char\* new\_name);

Void update\_position(Worker\* worker, Char\* new\_position);

Void update\_salary(Worker\* worker, Double new\_salary);

Void assign\_to\_department(Worker\* worker, Char\* new\_department);

Void assign\_to\_factory(Worker\* worker, Int new\_factory\_id);

Void display\_worker\_info(Worker\* worker);

Struct CarComponent:

Int component\_id

Char\* name

Char\* type

Double cost

Functions:

Void update\_component\_name(CarComponent\* component, Char\* new\_name);

Void update\_component\_type(CarComponent\* component, Char\* new\_type);

Void update\_cost(CarComponent\* component, Double new\_cost);

Void display\_component\_info(CarComponent\* component);

פונקציות לשליטה במערכת

// Upload system data from file

void upload\_system\_data(Company\*\* company);

// write system data to file

void write\_system\_data(Company\* company);

// Start the company system

Company\* start\_company\_system();

void run\_company\_system(Company\*\* company);

// Display functions

void display\_company(Company\* company);

void display\_distribution\_networks(Company\* company);

void display\_shops(Company\* company);

void display\_factories(Company\* company);

void display\_car\_models(Company\* company);

// Add new entities

void change\_CEO\_to\_sys(Company\* company);

void add\_worker\_to\_sys(Company\* company); // Adds a new worker based on user input

void add\_shop\_to\_sys(Company\* company); // Adds a new shop based on user input

void add\_factory\_to\_sys(Company\* company); // Adds a new factory based on user input

void add\_car\_model\_to\_sys(Company\* company); // Adds a new car model based on user input

void add\_component\_to\_sys(Company\* company);

void add\_model\_to\_inventory\_sys(Company\* company);

// Sorting workers

void sort\_workers\_by\_name(Company\* company);

void sort\_workers\_by\_id(Company\* company);

void sort\_workers\_by\_job\_details(Company\* company);

// Searching workers

void search\_worker\_by\_name(Company\* company);

void search\_worker\_by\_id(Company\* company);

void search\_worker\_by\_job\_detail(Company\* company);

// move worker from factory/shop to factory/shop

void move\_worker\_factory\_shop(Company\* company);

// Fire worker from the company

void fire\_worker\_from\_sys(Company\* company);