DATABASE PROJECT

By: Sreenavya (1978635)
Deeksha Rao (1970751)
Simerpreet Kaur (1970697)

Retail Application

- Considered raw real-time data set
- Decomposed through Normalization
- Python based web-application using FLASK
- Database backend on AWS Cloud
- Frontend: CSS, HTML, Java script

Raw Data → Normalized Data

Raw Data Set

Original(InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country)

After Normalization

Customer(CustomerID INT NOT NULL, Country VARCHAR(150),PRIMARY KEY(CustomerID));

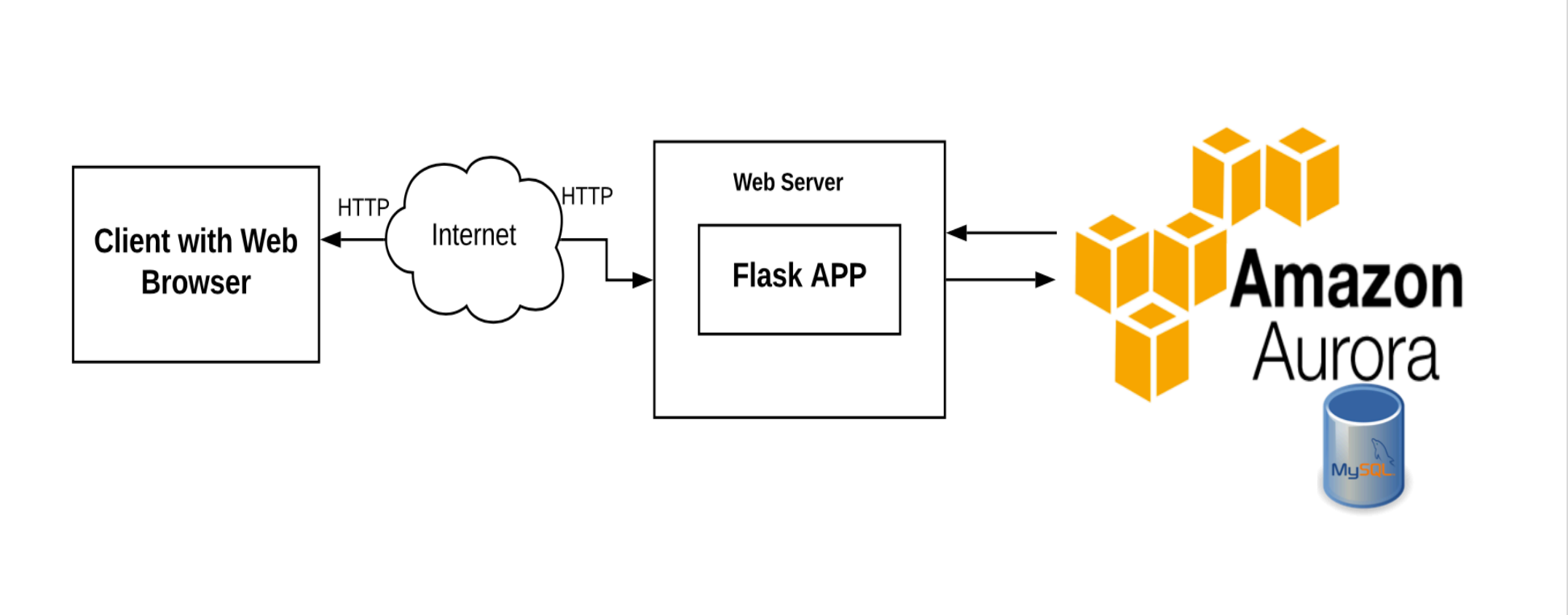
InvoiceDetails(InvoiceNo INT NOT NULL, StockCode VARCHAR(7), Quantity INT, PRIMARY KEY(InvoiceNo),FOREIGN KEY(StockCode) REFERENCES Item(StockCode));

Customer_Invoice (InvoiceNo INT, CustomerID INT, FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID), FOREIGN KEY(InvoiceNo) REFERENCES InvoiceDetails(InvoiceNo));

Invoice(InvoiceNo INT, InvoiceDate DATETIME, FOREIGN KEY(InvoiceNo) REFERENCES InvoiceDetails(InvoiceNo);

Item(StockCode VARCHAR(7) NOT NULL, Description VARCHAR(150) NOT NULL, UnitPrice FLOAT, PRIMARY KEY(StockCode));

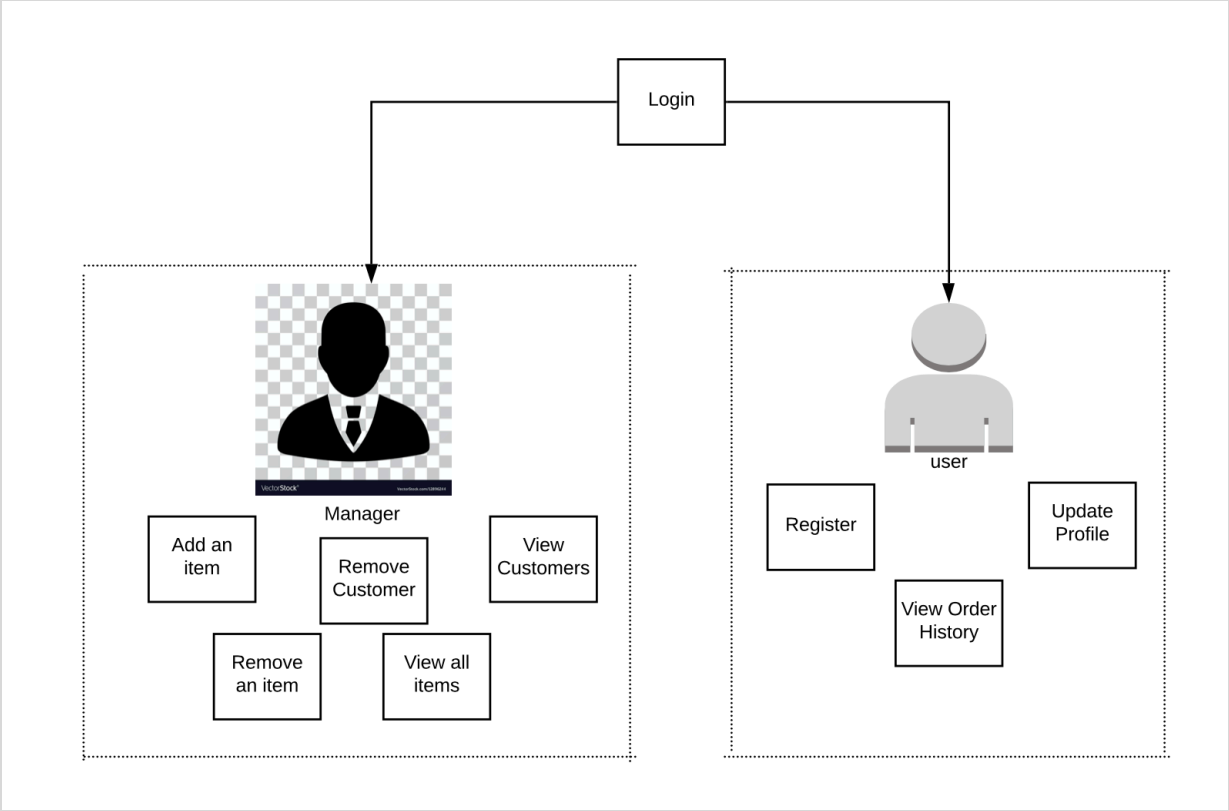
Back end design



Implemented Functionalities

- **DDL** : Create
- **DML**: Insert, Update, Delete
- **Stored Procedure**: Customer Registration and Adding an Item
- **Trigger:** To maintain new Customer Data on Registration
- **Cursor:** To execute SQL statements from FLASK
- **Cascade:** Deletion of customer cascades delete on Customer_Invoice
- **View:** Creating a view on Customer Table
  Commit: Changes are updated in the DB using commit

Use Case Diagram



CONTRIBUTIONS

SREENAVYA	SIMERPREET	DEEKSHA
Normalization	Normalization	Normalization
Documentation	Documentation	Documentation
Raw Data Set Selection	Customer Session	Stored Procedures
ER Diagram	Manager Session	Triggers
Application Design and Integration	Cascade	Cloud DB
Update		Cursor
Views		

DEMO

Conclusion

- Normalized a Raw Real Time Data Set
- Created the Database on Amazon Aurora DB
- Enforced Key constraints and Integrity constraints on the database
- Functionalities implemented-

Database Side
DDL (Create)
DML (Insert, Update)
Cascade
Triggers
Stored Procedures
Cursor
Views

Web Application Side
Session Management
User Separation

References

- https://medium.com/@simranjitkamboj/creating-a-web-application-using-python-flask-with-server-side-rendering-9ebea8204193
- https://danidee10.github.io/2016/10/05/flask-by-example-5.html


thank
you