**Exercise: Enhanced Bank Account Management with File Operations**

Create a Python program that simulates an enhanced bank account management system with the capability to save and load account data to/from a text file. Implement a class named **BankAccount** with the following features:

1. The class should have attributes:
    - **account_number** (string): A unique account number.(if the account exists, print message to user that account already exists)
    - **account_holder** (string): The name of the account holder.
    - **balance** (float): The current balance of the account.
2. Implement methods:
    - **__init__(self, account_number, account_holder, initial_balance)**: Initializes a new bank account with the provided account number, account holder name, and initial balance.
    - **deposit(self, amount)**: Adds the specified amount to the account balance. Ensure that the deposit amount is positive.
    - **withdraw(self, amount)**: Subtracts the specified amount from the account balance. Ensure that the withdrawal amount is positive and does not result in a negative balance.
    - **get_balance(self)**: Returns the current balance of the account.
    - **transfer(self, target_account, amount)**: Transfers funds from the current account to a target account. Ensure that the transfer amount is positive, and the current account has sufficient funds.
    - **save_to_file(self, filename="bank_accounts.txt")**: Saves the account data to a text file. Each line in the file should represent an account with the format **account_number,account_holder,balance**.
    - **load_from_file(filename="bank_accounts.txt")**: Loads account data from a text file and returns a list of **BankAccount** instances.
3. Create a separate function named **update_account_data(account1, account2)** that demonstrates how to use the **save_to_file** and **load_from_file** methods to update the data in the file. The function should save the account data to a file, load it back, and display the loaded account data for verification.

**Example Usage:**

```
# Create Bank Accounts
account1 = BankAccount("123456", "Alice", 1000.0)
account2 = BankAccount("789012", "Bob", 500.0)

# Perform Transactions
account1.deposit(500.0)
account1.withdraw(200.0)

account2.deposit(300.0)

account1.transfer(account2, 150.0)

# Display Current Balances
print(f"\nCurrent Balance for {account1.account_holder}'s account: ${account1.get_balance()}")
print(f"Current Balance for {account2.account_holder}'s account: ${account2.get_balance()}")

# Update Account Data in a File
update_account_data(account1, account2)
```