As part of the VideoShare web application, a database needs to be configured to store user, course, and video information.

The following steps detail how to set up the database, with some options for customization.

Step 1 - Install Sql Server and create the database

The database uses Microsoft Sql Server 2022 Express Edition (found here), although any version of Sql Server should be sufficient. When installing, make sure the database is configured to “Allow Remote Connections”, “Enable FILESTREAM for Transact-SQL access”, and “Enable FILESTREAM for File I/O Streaming Access”.

As part of installing SQL server, you will need to create a database account with administrative permissions. This is the account you will use for creating and managing the VideoShare database.

Next, create a new database for use by the web application. The database can be named as desired, but for the purposes of these instructions, it will be referred to as “WebAppDb”. Any future steps that use “WebAppDb” can be replaced with the name of your database.

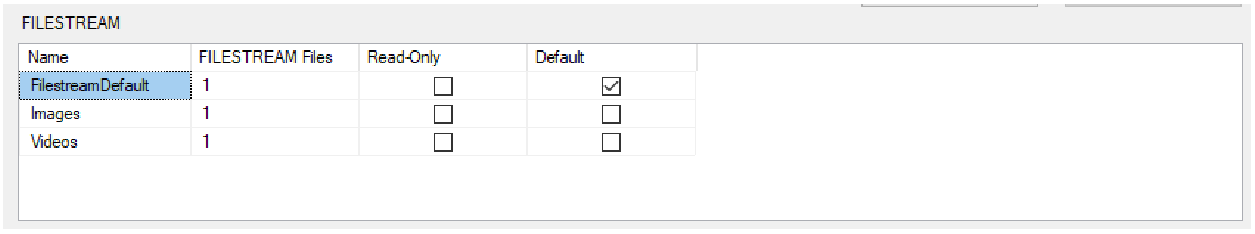
Step 2 - Configure the database to use FILESTREAM

Once the database has been created, a few additional steps need to be taken to allow for FILESTREAM data types (which is what the application uses to store photos, videos, and attachments).

In Sql Server Management Studio, open the WebAppDb database properties by right clicking on the database, then navigate to the Filegroups tab. There should be a section labeled FILESTREAM with no filegroups created. Create the following three filegroups:

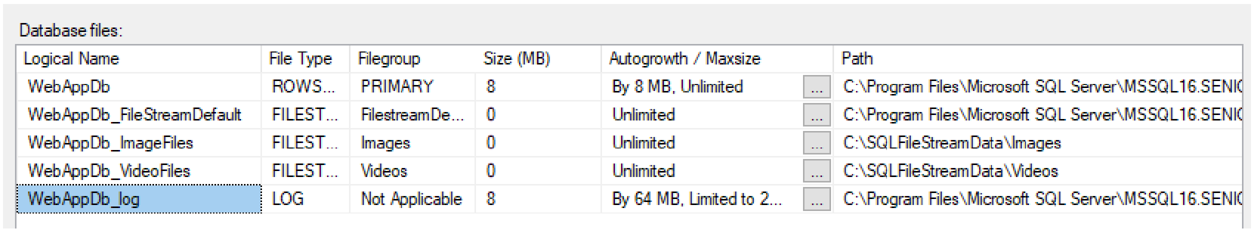
FilestreamDefault
Images
Videos

Make sure that FilestreamDefault is set as the default filegroup for FILESTREAM columns. When finished, the settings window should look like this:


FILESTREAM

Name	FILESTREAM Files	Read-Only	Default
FilestreamDefault	1	☐	✓
Images	1	☐	☐
Videos	1	☐	☐

Once the filegroups have been created, they need to be assigned to a file location. Navigate to Database Properties > Files, and create three new database files - named WebAppDb_FileStreamDefault, WebAppDb_ImageFiles, and WebAppDb_VideoFiles. These should have a file type of FILESTREAM, and their filegroups should be the file groups that were created in the previous step. Set up the file path in which you would like your files to be located, then press OK to save the configuration. The file path should use the same directory for Images and Videos, so that a network share can be created later for the web application. Your screen should look similar to this:



Database files:

Logical Name	File Type	Filegroup	Size (MB)	Autogrowth / Maxsize		Path
WebAppDb	ROWS...	PRIMARY	8	By 8 MB, Unlimited	...	C:\Program Files\Microsoft SQL Server\MSSQL16.SENIC
WebAppDb_FileStreamDefault	FILEST...	FilestreamDe...	0	Unlimited	...	C:\Program Files\Microsoft SQL Server\MSSQL16.SENIC
WebAppDb_ImageFiles	FILEST...	Images	0	Unlimited	...	C:\SQLFileStreamData\Images
WebAppDb_VideoFiles	FILEST...	Videos	0	Unlimited	...	C:\SQLFileStreamData\Videos
WebAppDb_log	LOG	Not Applicable	8	By 64 MB, Limited to 2...	...	C:\Program Files\Microsoft SQL Server\MSSQL16.SENIC

Note: If an error window pops up saying that the database is not configured for Always On Availability groups, that is a known issue with SQL Server 2022. It was fixed in Cumulative Update 1 for SQL Server 2022, released in February 2023, found here.

If additional help is needed configuring FILESTREAMs for access, check the Microsoft documentation here.

Step 3 - Create the application user

In order to better secure the code, a specific database account should be created for the application. For the purposes of this guide, the account will be named “appuser”, but you may rename it if preferred.

The account should have a SQL Server login, with a password defined by you. In account properties: Under General, the default database should be “WebAppDb”. Under User Mapping, the account should be mapped to a login on WebAppDb, with the username “appuser” and default schema of dbo. Under Status, the account should have permission to connect to the database engine set to “Grant”, and Login set to “Enabled”.

Step 4 - Create and populate the database tables

Now that the database has been created and configured, you need to create the tables that the application uses. Execute the following Database SQL Queries from the VideoShare GitHub repository IN ORDER:

- DDL/…
  - User Table.sql
  - Course Table.sql

    - Video Table.sql
    - Attachment Table.sql
    - Message Table.sql
    - UserxCourse Table.sql
    - UserxVideo Table.sql
    - Completion Percentage UDF.sql
    - Course Most Recent Visit UDF.sql
    - Views.sql
- DCL/…
    - appuser Permissions.sql
    - NOTE: This provides the application user with the necessary access to the database schema. If you are an experienced DBA and wish to use your own permissions for the appuser, you are encouraged to do so. Keep in mind that it may cause some access issues for the application if done incorrectly

Although the application can be run from scratch, there is also some test data included in the repository to get you started. In order to use the test data, run this query:

DML/Test Data Full.sql

NOTE: This query relies on the database having had no prior users, courses, or videos created. If that is not the case, you will need to run the queries individually and update the IDs of the test items within them.

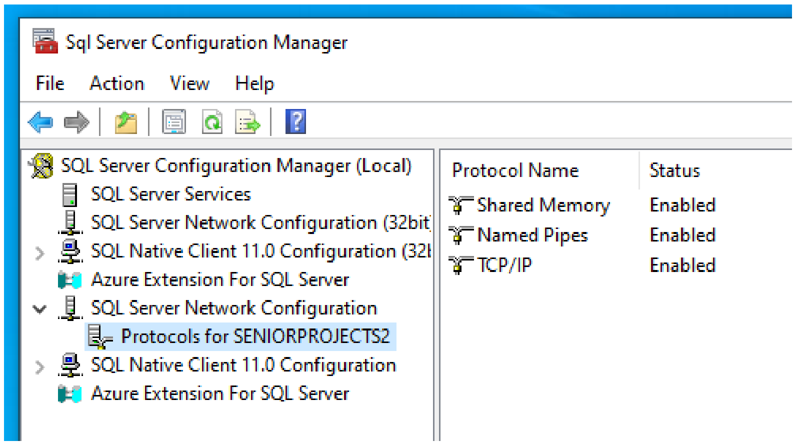
The test users in the database are:

Email Address	Password	Name	User Type + Notes
fakeemail@email.com	“Password”	Test One	Admin - Owns Course 1
anotherfake@email.com	“Password”	Test Two	Standard - This user is in every course
alsofake@email.com	“Password”	Test Three	Standard - Owns Courses 2 & 3
asdf@email.com	“Password”	Test Four	Standard
noemail@email.com	“Password”	Test Five	Standard - Has not joined any courses

Step 5 - Enable remote database connection

In order for the application to connect to the database via a connection string, the database server needs to be set up to allow access for the application user. Depending on your network configuration, this can require different setup, but for a simple computer-to-computer connection, the following settings must be changed:

In Sql Server Configuration Manager, for your database instance (not WebAppDb, but the entire server instance), ensure that, under Protocols, TCP/IP is set to Enabled

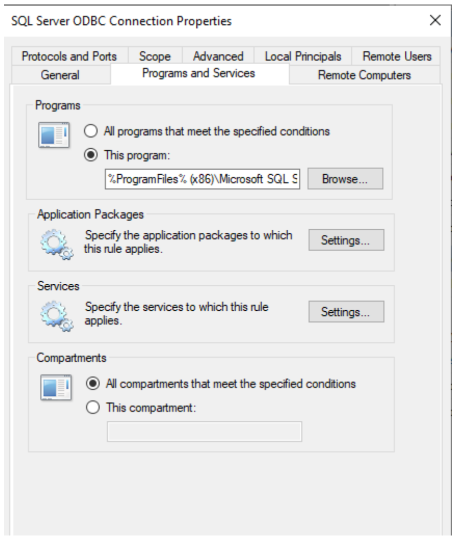


SQL Server Configuration Manager (Local)
- SQL Server Services
- SQL Server Network Configuration (32bit
- SQL Native Client 11.0 Configuration (32l
- Azure Extension For SQL Server
- SQL Server Network Configuration
  - Protocols for SENIORPROJECTS2
- SQL Native Client 11.0 Configuration
- Azure Extension For SQL Server

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Enabled
TCP/IP	Enabled

<u>The following steps are necessary if you are connecting to the database from a different computer.</u>

Under Windows Firewall, create an Inbound Rule allowing the program sqlbrowser.exe to execute, and allow any protocol, all ports. This restriction can be tightened as needed if you have experience with network security. Sqlbrowser.exe is typically found under Program Files(x86)\Microsoft SQL Server\90\Shared. The following screenshot shows the rule properties pane when it has been correctly configured.



Programs

All programs that meet the specified conditions [✓ This program:]

%ProgramFiles% (x86)\Microsoft SQL S

Application Packages

Specify the application packages to which this rule applies.

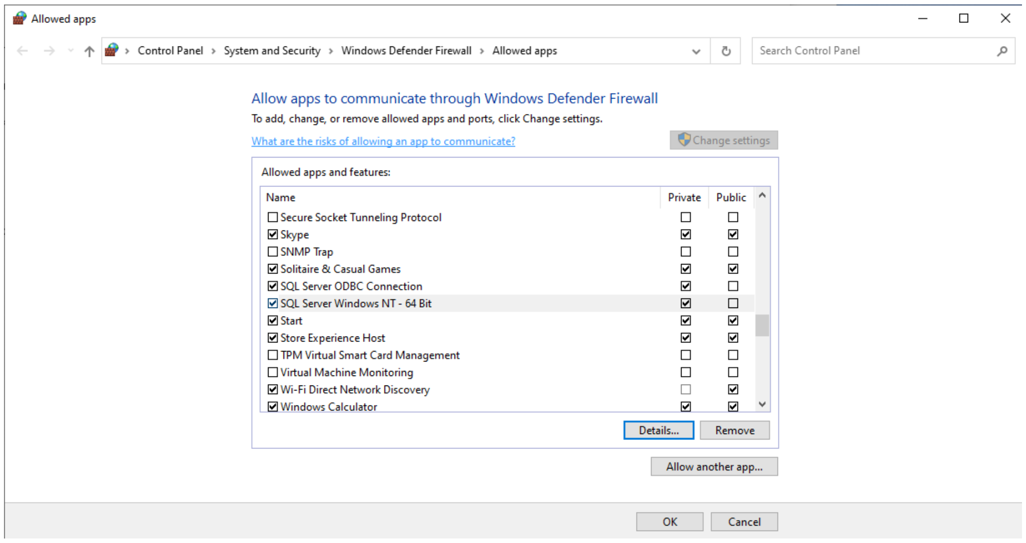
Services

Specify the services to which this rule applies.

Compartments

[✓ All compartments that meet the specified conditions] This compartment:

In Windows settings, under Update & Security, navigate to “Windows Security” and open Windows Security. Under firewall & network protection, go to “Allow an app through firewall”. Select Change Settings, then “add another app”. Add the filepath to sqlbrowser.exe, then save changes. Make sure that the network profile matches the profile of the network that you are using to connect to the database server.



Allowed apps

Control Panel › System and Security › Windows Defender Firewall › Allowed apps

Search Control Panel

Allow apps to communicate through Windows Defender Firewall

To add, change, or remove allowed apps and ports, click Change settings.

What are the risks of allowing an app to communicate?

Change settings

Allowed apps and features:

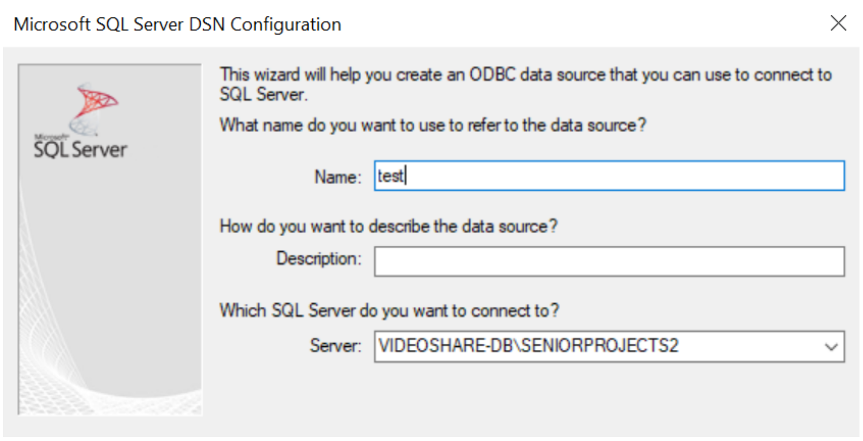
Name	Private	Public
☐ Secure Socket Tunneling Protocol	☐	☐
☑ Skype	☑	☑
☐ SNMP Trap	☐	☐
☑ Solitaire & Casual Games	☑	☑
☑ SQL Server ODBC Connection	☑	☐
☑ SQL Server Windows NT - 64 Bit	☑	☐
☑ Start	☑	☑
☑ Store Experience Host	☑	☑
☐ TPM Virtual Smart Card Management	☐	☐
☐ Virtual Machine Monitoring	☐	☐
☑ Wi-Fi Direct Network Discovery	☐	☑
☑ Windows Calculator	☑	☑

If there are additional issues with your network, you will need to troubleshoot those individually to ensure that the web application can access the database. Additional resources can be found on the Microsoft website, here.

Step 6 - Create and test a connection string for the web server environment

Once the network has been configured, a connection needs to be established between the application environment and the database environment. To test this manually, you can use an ODBC connection to connect to the database as the appuser account.

Search your windows programs for "ODBC", and open the ODBC data source administrator. Under User DSNs, select "Add…" to create a new test connection. Ensure that you have the correct ODBC driver installed (ODBC Driver 17 or 18 for SQL Server worked for me), then create the connection. You will need to enter a name, description, and network location for the SQL Server instance.



Microsoft SQL Server DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: test

How do you want to describe the data source?

Description:

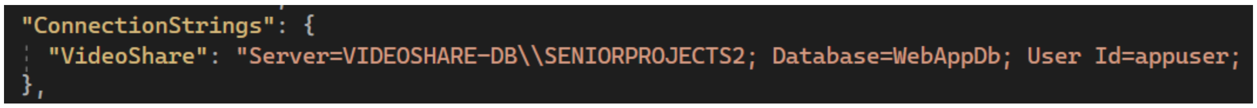
Which SQL Server do you want to connect to?

Server: VIDEOSHARE-DB\SENIORPROJECTS2

In the next step, select the authentication method for appuser - SQL Server login by username and password - and enter the database account username and password. Next, change the default database to WebAppDb. You can choose to configure additional settings for network security - this application uses an encrypted connection, so Connection Encryption can be set to Mandatory. If the database server has a valid TLS certificate, enter the hostname. Otherwise, just select “Trust server certificate”. When finished, select Test Datasource to attempt to connect to the database.

If connection is unsuccessful, then one of the previous network security steps may have not been completed, or additional troubleshooting is needed.

If successful, then the information used can be applied to the application through a connection string. This string should be stored in the application’s appsettings.json file under the name “VideoShare”.

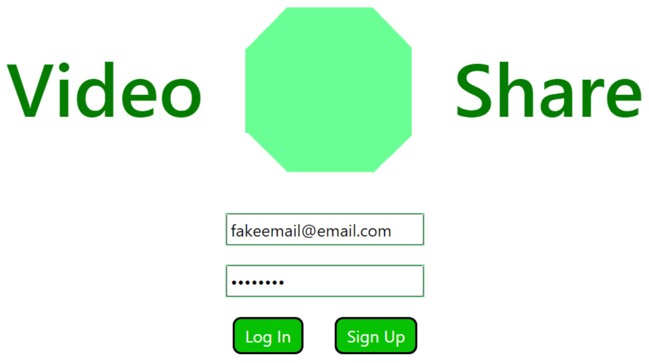


```
"ConnectionStrings": {
  "VideoShare": "Server=VIDEOSHARE-DB\\SENIORPROJECTS2; Database=WebAppDb; User Id=appuser;
},
```

For the connection string, use the following format:
“Server=[Server Name]; Database=WebAppDb; User Id=appuser; Password=[Password]; Encrypt=True; TrustServerCertificate=True;”

Server Name - The network path to your SQL Server instance. For me, this was VIDEOSHARE-DB\SENIORPROJECTS2. (Note: for JSON files, the backslash character needs to be escaped with an additional backslash. i.e. “VIDEOSHARE-DB\\SENIORPROJECTS2”)
Database - The name of the particular database to use.
Password - The password defined for appuser

Once this string has been configured in the app settings, setup has been completed, and the application should be able to connect to the database. This can be verified by running the application and attempting to log in as the test user “fakeemail@email.com”, with a password of “Password”. If you chose not to create test data, then the database can instead be tested by attempting to Sign Up on the login page.


Video
Share
fakeemail@email.com
••••••••
Log In
Sign Up