Use Cases:

- **Adding license.** A request is received by the API, which strips the security token from the request and sends it to the Authorization Server. The server validates their credentials and sends back a positive response. If the user is a manager, then the backend calls the database to add a new record using the fields provided by the user.
- **Updating license.** A request is received by the API, which strips the security token from the request and sends it to the Authorization Server. The server validates their credentials and sends back a positive response. If the user is a manager, then the backend calls the database to update the specified record using the fields provided by the user.
- **Removing license.** A request is received by the API, which strips the security token from the request and sends it to the Authorization Server. The server validates their credentials and sends back a positive response. If the user is a manager, then the backend calls the database to remove the specified record.
- **View all licenses**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database. The database response is returned to the user.
- **View licenses by license type**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses of the user-specified type. The database response is returned to the user.
- **View licenses by cost**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which

validates their credentials and sends back a positive response. The backend queries the database for all licenses of the user-specified cost. The database response is returned to the user.

- **View licenses by assigned device**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses assigned to the user-specified device. The database response is returned to the user.
- **View licenses by assigned employee**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses assigned to the user-specified employee. The database response is returned to the user.
- **View licenses by license ID**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses of the user-specified License ID. The database response is returned to the user.
- **View licenses by date of addition**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses of the user-specified addition date. The database response is returned to the user.

- **View licenses by date of expiration**. A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. The backend queries the database for all licenses of the user-specified expiration date. The database response is returned to the user.
- **Assign a license.** A request is received by the API, which then extracts their security token from the request. This token is sent to the Authorization Server, which validates their credentials and sends back a positive response. If the user is a manager then the backend calls the database to add a record linking the user-provided employee or device to the specified license.
- **Unauthorized access.** The API receives a request from a user outside the organization. The request will not have a security token attached, so the API will reject the communication attempt.
- **Invalid authorization token.** The API receives a request from a user with an invalid authorization token. The API will strip out their security token and send it to the authorization server. The authorization server will send back a negative response, so the API will reject the communication attempt.
- **Employee attempts to user manager-level actions.** The API receives a request from an employee. The application will strip their security token from the request and send it to the authorization server. The authorization server will send a positive response to the API including the user's credentials. The API will see that the user is not a manager and reject the communication attempt.

- **Database recovery.** A rogue manager has deleted a large number of records maliciously. A database administrator restores the most recent of the system's database backups. The DBA checks the log of the database to see who deleted the records.
- **Attempted DDoSing.** A malicious actor attempts to flood the API with requests. The API performs validity checking on the authorization token and all fields included with the requests. This validity checking drops most requests, preventing the authorization server or database, both integral to the system, from going down or experiencing delays.