Overview

ScheduleNow will be sectioned into a presentation layer, service layer and a data layer. The service layer will be created with Express.js. The simplicity and flexibility of Express.js will fit the routing, request and response needs of the application. Express.js will be used to create a RESTful API that will be facilitate communication between the client and the database. The application's business logic will be held in the service layer. This document details application endpoints, requests, responses and diagrams showing the communication between the UI pages and service endpoints.

Account

1. /login
   a. POST: Log into account
   When employees click the 'login' button (front-end), the application will authenticate the credentials stored in the database, create a new session and log in the employee or deny the credentials. There are no changes to employee or account data.

   Request

   ```
   POST /login
   {
       "username":"abrown",
       "password":"temp_password"
   }
   ```

   Response

   ```
   Success
   200 Ok
   {
       "message": "Login successful",
       "session_token":"TOKEN"
   }

   Error
   401 Unauthorized
   {
     "error": {
   ```

```
  "code": 401,
  "message": " Unauthorized "
 }
}
```

Diagram

‘Login’ page. ←→ API ←→ Database

2. /logout
    a. POST: Log out of account

When employees click the logout button (front-end) it will end the session and log out the employee. There are no changes to employee or account data.

Request

POST /logout
Host: www.swdv691-project.com
Authorization: TOKEN

Response

Success
200 Ok

```
{
    "message": "Logout successful"
}
```

Error
400 Bad Request

```
{
 "error": {
  "code": 400,
  "message": " Invalid session token "
 }
}
```

Diagram

‘Home’ page. ←→ API

3. /registration
    a. POST: Create an account

    When employees click the submit button on the registration page, an employee and an account Is created. Employee data and account data will be stored in the database.

Request

```
POST /registration
{
    "firstName":"Alice",
    "lastName":"Brown",
    "email":"abrown@email.com",
    "isManager":"true",
    "username":"abrown",
    "password":"temp_password"
}
```

Response

```
Success
201 Created
{
  "message": "Employee and account created successfully",
  "employee": {
    "id": "1010101010101",
    "firstName":"Alice",
    "lastName":"Brown",
    "email":"abrown@email.com",
    "isManager":"true",
    "accountId": "1212121212"
  },
“account”:{
    "accountId": "1212121212",
    "username": "abrown",
    “password”: “------",
    “isActive”: “true”
}
}
```

Error

If the request was sent without a name value.
400 Bad Request

```
{
 "error": {
  "code": 400,
  "message": " Invalid input data: Missing required field 'name'"
 }
}
```

Diagram

‘Registration’ page. ←→ API←→Database

4. /account/home
   a. GET: Get account home page

After an employee account is authenticated from the login page, a GET request to this endpoint is sent, a list of the employee’s appointments and availability are retrieved from the database and displayed to the front-end. There are no changes to employee or account data

Request

```
GET /account/home
Host: www.swdv691-project.com
Authentication: TOKEN
```

Response

Success
200 Ok

```
{
    "appointments":[
    {
            "id":"342",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "cancelled",
            "employeeId": "123",
            "customerId: "565"
```

```
        },{

                "id":"342",
                "date": "2025-01-02",
                “startTime”:”10:00AM”,
                “endTime”:”11:30AM”,
                "status": "scheduled",
                "employeeId": "123",
                "customerId: "568"

        }

        ],
        "availability":[
        {
                "day":"Monday"
                "startTime":"10:00 AM",
                "endTime":"2:00 PM",
                "employee_id": "123"
        }
        ]
}
```

Error

401 Unauthorized

```
{
  "error": {
    "code": 401,
    "message": " Invalid session token "
  }
}
```

Diagram

‘Home’ page. ←→ API←→Database

5. /account/password
    a. POST: Update account password

When employees click the submit button on the ‘change password’ page, the employee account password is updated in the account document for that employee. Account data is changed.

Request

```
POST /account/password
{
    "password":"abcdefg",
}
```

Response

Success
200 Ok

```
{
  "message": "Password reset is successful",
}
```

Error
400 Bad Request

```
{
  "error": {
    "code": 400,
    "message": " Invalid input data: Missing required field 'password'"
  }
}
```

Diagram

‘Password Change’ page. ←→ API←→Database

Employees

1. /employees/{employee id}/appointments
   a. GET: Get list of appointments for an employee
   When an employee logs into their account a list of their appointments are retrieved from the database and displayed on the front-end. There are no changes to employee or appointment data.

Request

```
GET employees/{employee id}/appointments
Host: www.swdv691-project.com
```

Response

Success

200 Ok

```
[
    {id":"342",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "cancelled",
            "employeeId": "{employee id}",
            "customerId: "565"

},{
            id":"343",
            "date": "2025-01-02",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "completed",
            "employeeId": "{employee id}",
            "customerId: "568"
    }
]
```

Error

404 Not Found

```
{
 "error": {
  "code": 404,
  "message": "Employee not found"
 }
}
```

Diagram

‘Home” page. ←→ API←→Database

2. /employees/{employee id}/availability

    a. GET: Get list of availability for an employee

    When an employee logs into their account a list of their availability are retrieved from the database and displayed on the front-end. There are no changes to employee or appointment data.

Request

GET employees/{employee id}/ availability
Host: www.swdv691-project.com

Response

Success
200 Ok

```
   {
         "day":"Monday"
         "startTime":"10:00 AM",
         "endTime":"2:00 PM",
         "employee_id": "{employee id}"
   }
```

Error
404 Not Found

```
{
 "error": {
  "code": 404,
  "message": "Employee not found"
 }
}
```

Diagram

‘Home” page. ←→ API←→Database

Appointments

1. /appointments

    a. GET: Get a list of appointments

When managers log into their accounts a list of all appointments are displayed on the home page. There are no changes to appointment data.

Request

```
GET /appointments
Host: www.swdv691-project.com
```

Response

Success

```
200 Ok
[
    {
            "id":"342",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "cancelled",
            "employeeId": "123",
            "customerId: "565"

    },{
            "id":"343",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "scheduled",
            "employeeId": "125",
            "customerId: "490"

    }
]
```

Error

```
401 Unauthorized
{
 "error": {
  "code": 401,
  "message": " Invalid session token "
 }
}
```

Diagram

‘Home’ page. ←→ API←→Database

b. POST: Create a new appointment

When employees click on the ‘submit’ button from the add appointment page, the appointment information from the front-end is captured and stored in the database. If customer information is entered on the add appointment page the information is captured and stored in the database. There are changes to the appointment collection and/or the customer collection.

Request

```
POST /appointments
{
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "scheduled",
            "employeeId": "123",
            "customerId: "565"

    }
```

Response

```
Success
201 Created
{

            "id":"342",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "scheduled",
            "employeeId": "123",
            "customerId: "565"


}
```

Error

If the request was sent without a date value.

400 Bad Request

```
{
 "error": {
  "code": 400,
  "message": " Invalid input data: Missing required field 'date'"
 }
}
```

Diagram

‘Add Appointment’ page. ←→ API←→Database

2. /appointments/{appointment id}
    a. GET: Get appointment details

When an employee selects an appointment and clicks the ‘edit’ button on the home page, appointment data is retrieved from the database and displayed on the front-end. There are no changes to appointment data.

Request

GET /appointments/{appointment id}
Host: www.swdv691-project.com

Response

Success

200 Ok

```
{
            "id":"{ appointment id}",
            "date": "2025-01-01",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "cancelled",
            "employeeId": "123",
            "customerId: "565"

    }
```

Error

```
404 Not Found
{
 "error": {
  "code": 404,
  "message": "Appointment not found"
 }
}
```

Diagram

‘Home’ page. ←→ API←→Database

b. PUT: Update appointment details

When an employee clicks the ‘submit’ button from the edit appointment page, appointment data is from the front-end is captured and stored in the database. There are changes to the appointment document.

Request

```
PUT /appointments/{appointment id}
{
            "date": "2025-01-23",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "scheduled",
            "employeeId": "123",
            "customerId: "565"

    }
```

Response

Success

```
200 Ok
{

            "id":"{ appointment id}",
            "date": "2025-01-23",
            “startTime”:”09:00AM”,
            “endTime”:”09:30AM”,
            "status": "scheduled",
            "employeeId": "123",
            "customerId: "565"

    }
```

Error
If the request was sent without a date value.
400 Bad Request

```
{
  "error": {
    "code": 400,
    "message": " Invalid input data: Missing required field 'date'"
  }
}
```

Diagram

‘Edit Appointment’ page. ←→ API←→Database

3. /appointments/search
   a. GET: Get a list of appointments based on search string

When an employee wants to find an appointment from a list located on the home page, an appointment can be searched using appointment details. Managers can search appointments for all employees. Regular employees can only search their appointments. There are no changes to appointment data.

Request

GET /appointments/search?status=Rescheduled
Host: www.swdv691-project.com

Response

Example as a manager
Success
200 Ok

```
[
{
            “_id”: “12345”,
            "date": "2025-01-01 09:00:00",
            "status": "Rescheduled",
            "employee_id": “33333”,
            "customer_id: "56565656"

    },
    {
            “_id”: “90000”,
            "date": "2025-01-15 13:00:00",
            "status": "Rescheduled",
            "employee_id": “444444”,
            "customer_id: "5000000"

    }
    ]
```

Error

400 Bad Request

```
{
  "error": {
    "code": 400,
    "message": " Search query is incomplete "
  }
}
```

Diagram

‘Home” page. ←→ API←→Database

Availability

1. /availability
   a. GET: Get list of availability

   When employees log into their accounts a list of all their availability are displayed on the home page. When managers log in they can see the availability of all employees. There are no changes to availability data.

   Request

   GET /availability/
   Host: www.swdv691-project.com

Response

Example as an employee
Success
200 Ok

```
[
    {
            "day of the week":"Monday"
            "start":"10:00 AM",
            "end":"2:00 PM",
            "employee_id": "12345"
    },
    {
            "day of the week":"Wednesday"
            "start":"1:00 PM",
            "end":"2:00 PM",
            "employee_id": "12345"
    },
    {
            "day of the week":"Friday"
            "start":"11:30 AM",
            "end":"2:00 PM",
            "employee_id": "12678"
    }

]
```

Error
401 Unauthorized

```
{
 "error": {
  "code": 401,
  "message": " Invalid session token "
 }
}
```

Diagram

‘Home” page. ←→ API←→Database

b. POST: Create new availability

When employees click on the 'submit' button from the add availability page, the information from the front-end is captured and stored in the database. There are changes to the availability collection.

Request

```
POST /availability/
Host: www.swdv691-project.com
{
        "day of the week":"Thursday",
        "start":"1:00 PM",
        "end":"3:00 PM",
}
```

Response

Success
200 Ok

```
{
        “_id”:”4567”,
        "day of the week":"Thursday",
        "start":"1:00 PM",
        "end":"3:00 PM",
        "employee_id": "12345"
}
```

Error
If the request was sent without a start value.
400 Bad Request

```
{
  "error": {
    "code": 400,
    "message": " Invalid input data: Missing required field 'start’'"
  }
}
```

Diagram

‘Add Availability” page. ←→ API←→Database

2. /availability/{availability id}
    a. GET: Get availability details

    When an employee selects an availability and clicks the 'edit' button on the home page, availability data is retrieved from the database and displayed on the front-end. There are no changes to availability data.

    Request

    GET /availability/{availability id}
    Host: www.swdv691-project.com

    Response

    Success
    200 Ok

```
{
            "_id": "{availability id}",
            "day of the week":"Thursday",
            "start":"1:00 PM",
            "end":"3:00 PM",
            "employee_id": "12345"
}
```

    Error
    404 Not Found

```
{
 "error": {
  "code": 404,
  "message": "Availability not found"
 }
}
```

    Diagram

    'Home' page. ←→ API←→Database

    b. PUT: Update availability details

    When an employee clicks the 'submit' button from the edit availability page, availability data from the front-end is captured and stored in the database. There are changes to the availability document.

Request

```
PUT /availability/{availability id}
{
            "day of the week":"Thursday”,
            "start":"6:00 PM",
            "end":"6:40 PM"
}
```

Response

Success
200 Ok

```
{
            “_id”: “{availability id}”,
            "day of the week":"Thursday”,
            "start":"6:00 PM",
            "end":"6:40 PM",
            "employee_id": "12345",
}
```

Error
If the request was sent without an end value.
400 Bad Request

```
{
  "error": {
    "code": 400,
    "message": " Invalid input data: Missing required field ‘end’"
  }
}
```

Diagram

‘Edit Appointment’ page. ←→ API←→Database

3. /availability/search
    a. GET: Get a list of availability based on search string

When a manager wants to find an availability from a list located on the home page, an availability can be searched using availability details. There are no changes to availability data.

Request

```
GET /availability/search?day_of_the_week=Monday
Host: www.swdv691-project.com
```

Response

```
Success
200 Ok
[
{
		“_id”:”90001”,
		"day_of_the_week":"Monday",
		"start":"1:00 PM",
		"end":"3:00 PM",
		"employee_id": "12345"
},
{
		“_id”:”90345”,
		"day of the week":"Monday",
		"start":"1:00 PM",
		"end":"3:00 PM",
		"employee_id": "12367"
}
]
```

```
Error
404 Not Found
{
 "error": {
  "code": 404,
  "message": "Availability not found"
 }
}
```

Diagram

‘Home’ page. ←→ API←→Database

Customers

1. /customers/search
    a. GET: Get customer details based on search string

    When an employee is on the ‘add/edit appointment page’, customer data can be found using the search feature. There are no changes to customer data.

    Request

    GET /customers/search?name=Brian
    Host: www.swdv691-project.com

    Response

    Success
    200 Ok

```
[{
            “_id”:”90001”,
            "name":"Brian Costa",
            "email":"bcosta@email.com",
            "street":"123 Main St.",
            "city": "New York City",
            "state": "NY"
},
{
            “_id”:”90034”,
            "name":"Brian Calloway",
            "email":"bcalloway@email.com",
            "street":"9785 Lake St.",
            "city": "New York City",
            "state": "NY"
}]
```

    Error
    404 Not Found

```
{
  "error": {
    "code": 404,
    "message": "Customer not found"
  }
}
```

Diagram

‘Add/Edit Appointment’ page. ←→ API←→Database