

# T3A2 - Stream-lined Form App

By James & Jit

# Website address

stream-lined.netlify.app

Access:

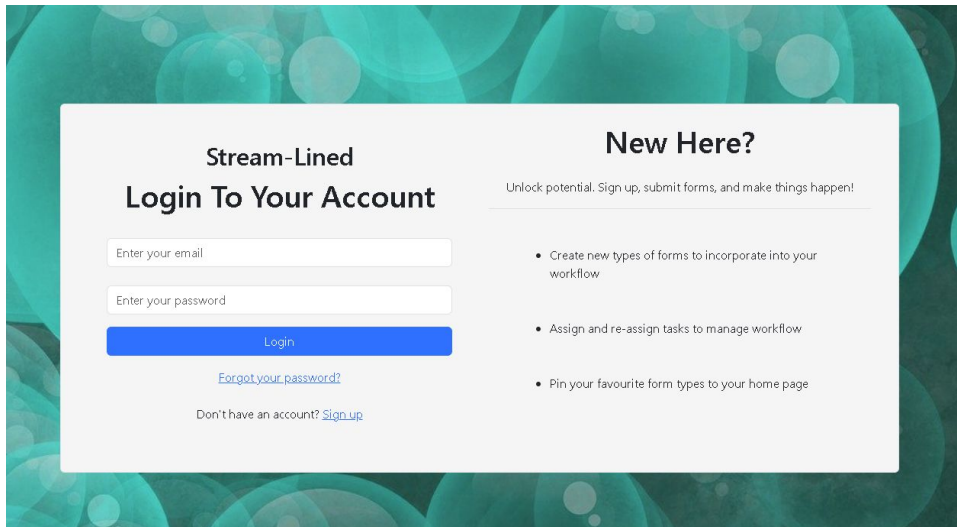
Email: [admin@email.com](mailto:admin@email.com)

Password: admin

or sign-up with your own email with  
limited access

Will discuss in a moment with account  
creation page

# Login



The desktop view of the Stream-Lined login page features a light gray background with a teal bokeh pattern. The page is divided into two main sections. The left section, titled "Stream-Lined Login To Your Account", contains two input fields for "Enter your email" and "Enter your password", a blue "Login" button, a link for "Forgot your password?", and a link for "Don't have an account? Sign up". The right section, titled "New Here?", includes a sub-header "Unlock potential. Sign up, submit forms, and make things happen!" and a list of three bullet points: "Create new types of forms to incorporate into your workflow", "Assign and re-assign tasks to manage workflow", and "Pin your favourite form types to your home page".

**Stream-Lined**  
**Login To Your Account**

Enter your email

Enter your password

Login

[Forgot your password?](#)

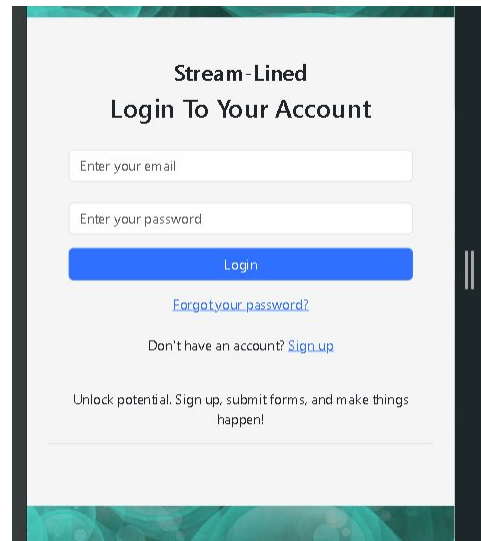
Don't have an account? [Sign up](#)

**New Here?**

Unlock potential. Sign up, submit forms, and make things happen!

- Create new types of forms to incorporate into your workflow
- Assign and re-assign tasks to manage workflow
- Pin your favourite form types to your home page

- Login via email and password
- Basic information about the app
- Navigate to signup if without account
- Auto-login with valid token (3 days)
- Responsive layout



The mobile view of the Stream-Lined login page is shown within a black frame. It features a light gray background with a teal bokeh pattern. The page is titled "Stream-Lined Login To Your Account". It contains two input fields for "Enter your email" and "Enter your password", a blue "Login" button, a link for "Forgot your password?", and a link for "Don't have an account? Sign up". The sub-header "Unlock potential. Sign up, submit forms, and make things happen!" is also present.

**Stream-Lined**  
**Login To Your Account**

Enter your email

Enter your password

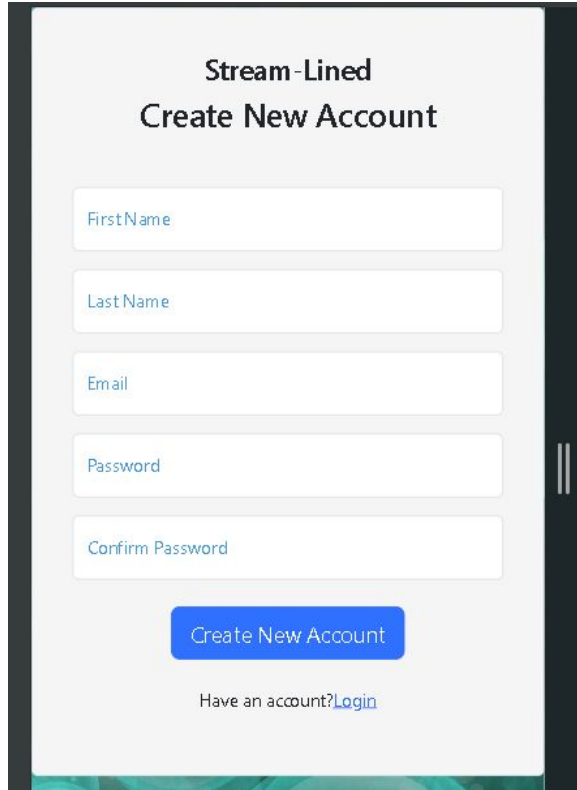
Login

[Forgot your password?](#)

Don't have an account? [Sign up](#)

Unlock potential. Sign up, submit forms, and make things happen!

# Account Creation

A mobile app interface for creating a new account. The form is titled "Stream-Lined Create New Account" and contains five input fields: "First Name", "Last Name", "Email", "Password", and "Confirm Password". A blue button labeled "Create New Account" is positioned below the fields. At the bottom, there is a link "Have an account? Login". The form is set against a light gray background with a dark gray border on the right side.

**Stream-Lined**  
**Create New Account**

First Name

Last Name

Email

Password

Confirm Password

Create New Account

Have an account? [Login](#)

- Enter basic user information
- Auto assigned user level access
- Creating a new account will take you to the Home page

# Home Page

Stream-Lined

Home

Form Builder

Forms

Actions

Logout

## Step by Step

1. Navigate to the forms page
2. Select the form you would like to submit
3. Complete and submit your form
4. Comment on forms assigned to you on the Actions Page

Quick step by step to guide new users

## Favourite Forms

### Trial Form

[Click here to fill out the Trial Form](#)

Quick access to any favoured form types

# Form Builder

Stream-Line

Home Form Builder Forms Actions Logout

### Form Components

Full Name

Email

Short Question and Answer

Long Question and Answer

File Upload

Date and Time

Reset

Form Name:  
Trial

Assigned To:  
Select a User

1  
Full Name:  

Remove

3  
Where were you born?  
Answer  

Remove

5  
Date and Time:  
dd/mm/yyyy --:-- --  

Remove

tester test

Remove

Preview Form

Save Template

- Select different components to add to the new form template
- Add a title and assign a user that will manage future submitted forms
- Some components are customizable e.g. short question and answer
- Buttons to remove elements or reset the entire template build in progress
- Save Template to finalize completion process
- Additional form components to be created with further development

# Form Builder - continuation

The screenshot shows a web application for building forms. On the left is a sidebar with a menu icon and links: Stream-Lined, Home, Form Builder (active), Forms, Actions, and Logout. Below the sidebar is a 'Form Components' panel listing: Full Name, Email, Short Question and Answer, Long Question and Answer, File Upload, and Date and Time. The main workspace contains a form with a 'Reset' button, a 'Form Name' input, an 'Assigned To' dropdown (showing 'Select a User'), and two numbered sections. Section 1 has an 'Email' input and a 'Remove' button. Section 2 has an 'Enter Question Here' input, a 'Save' button, and a 'Remove' button. At the bottom are 'Preview Form' and 'Save Template' buttons.

← Responsive layouts

Preview form before creating→

The screenshot shows a preview of a form in a modal window titled 'Overlay Title' with a close button (X). The form content includes: a large 'Trial' heading with a 'Favourite' checkbox, a 'Full Name' input, an 'Email' input, a 'Where were you born?' input (showing 'Answer'), and a 'Describe your childhood.' input with a text area. A blue 'Submit' button is at the bottom. A 'Close' button is in the bottom right corner of the modal.

# Form Page

- Navigate between different templates at anytime
- Submit new forms based on the selected templates
- See previous submission and select to view
- Delete/Edit the template if you have admin or manager access

Stream-Lined

HomeForm BuilderFormsActionsLogout

## Form Page

Form Name

test

123

<1>

test

☒ Favourite

Enter form description

Submit New Form

Edit

Delete Template

### Previously Submitted Forms

description	submission date/time	submitted by	status	assigned	tasked user
THIS IS A DESCRIPTIONNN!!!!	08-03-2024 / 7PM	admin test	open	admin	

<1>



# On 'Submit New Form'

- Fill out fields as per the created template
- Favourite the template for quick access
- Submit the form to the database for storage

Stream-Lined

Home Form Builder Forms Actions Logout

Form Page

Form Name

Test

< 1 >

Test ☒ Favourite

THIS IS A DESCRIPTIONNN!!!!

Full Name:

Email:

Do you enjoy tests?  
Answer

Please explain your answer

Submit

# Actions Page

Stream-Lined

Home Form Builder Forms Actions Logout

## Actions

assigned

### Test

Full Name:  
MYNAMEJIT

Email:  
justrandom@email.com

Do you enjoy tests?  
Super excited

Please explain your answer  
Just cause :)

Add Comment Here

All self created forms regardless of status can be viewed from the actions page via the forms table link

## Previously Submitted Forms

description	submission date/time	submitted by	status	assigned	tasked user
THIS IS A DESCRIPTIONNN!!!!	08-03-2024 / 7PM	admin test	open	admin	

# Actions - continuation

## Actions

tasked

test re-assignment

assigned

## Trial

Full Name:

George

did you eat the cookie?

Yes

explain why

Because I was told too by the man in my head

comment 1

Just a regular old comment

manager, 2024-03-08T09:53:11.663Z

Add Comment Here

Welcome to the end of the course!!!!!!!!!!!!!!

Request action from:

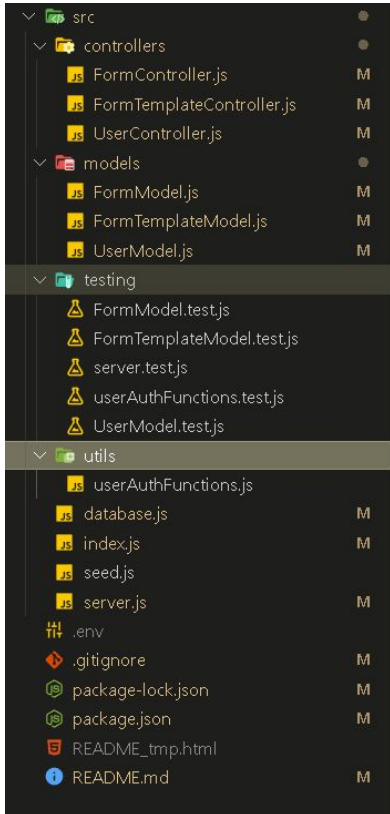
Select a User

Re-assign

Close Form

- Append comments and re-assign the form to different active users
- Close the form to prevent further commenting and removal from action queues
- Split queues for items assigned to the user and those tasked by other users for comment

# Code Backend



Node environment  
routes with the  
application using  
the expressJS  
framework

Other  
dependencies  
include bcryptjs,  
cors, dotenv,  
jsonwebtoken,  
mongoose

Structured layout  
keep functional  
components  
grouped together

```
router.get("/:id", async(request, response) => {
  try{
    let result = await User.findById(request.params.id);
    if(!result){
      return response.status(404).json({message:"User not found"});
    }
    return response.json(result);
  }catch(error){
    return response.status(500).json({message: " Internal server error"});
  }
})

//Add Favourites

router.patch('/favourites', async (request, response) => {

  try {
    const id = getUserIdFromToken(request.headers.jwt);
    if (!id) {
      return response.status(401).json({ error: 'Invalid token' });
    }

    const result = await User.findOneAndUpdate(
      { _id: id },
      { $set: { favourites: request.body.favourite } },
      { new: true }
    );

    if (!result) {
      return response.status(404).json({ error: 'User not found' });
    }
    const { favourites } = result.toObject()
    return response.json(favourites);
  } catch (error) {
    console.error('Error updating favourites:', error);
    return response.status(500).json({ error: 'Internal server error' });
  }
});
```

Extensive error handling  
while communicating  
asynchronously to the  
database

Example favourites patch  
route

- Uses auxiliary  
function to get the Id  
of the user from the  
json web token
- Finds the user and  
updates the  
favourites array in  
one swift motion

# Code Front-end

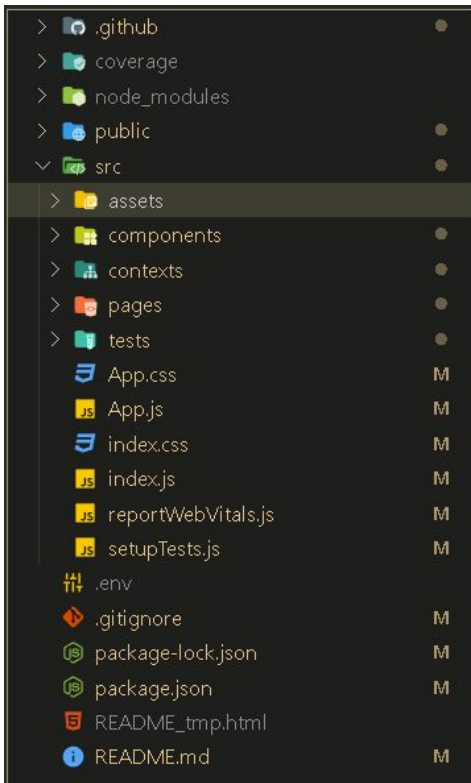
- Initialized with create-react-app
- Components organised by function
- Code structured with headers to compartmentalize functionality and easily find
- State manipulation, contexts, cookies, functions to handle user interactions, asynchronous database calls with error handling
- Variety of dependencies including, Antd - Responsive layouts, Moment - Manipulating/formatting dates, js-cookie - handling cookies, bootstrap - quick css design

```
export default function FillOutForm({formName, formDescription, set
//cookies
const jwt = Cookies.get('jwt')
//contexts
const {apiUrl} = useContext(ApiContext)
const {formComponents, formTemplate, fetchFormTemplate} = useC
//useStates
const [formData, setFormData] = useState({});

//handles
const handleInputChange = (index, value) => {
  setFormData(prevData => ({
    ...prevData,
    [index]: value
  }));
};

const handleSubmit = async (event) => {
  event.preventDefault()

  const form = {
    description: formDescription,
    formTemplate: formTemplate_id,
    formData: formData,
    assignedTo: formTemplate.assignedTo
  }
  try {
    const response = await fetch(`${apiUrl}/forms/submit`,
      method: 'POST',
```



# Testing

			✓	✗
			Testers: James	
Index	Test	Description	Result	Comm
1	Signup			
1.1		signup link connects to create a new user page	✓	
1.2		normal login process with accepted values works fine	✓	
1.3		create new account button navigates to home page with correct values	✓	
1.4		trying fields blank is rejected with a field specific helper message	✗	rejected
1.5		trying to use an email already in use is rejected, with helper message	✗	rejected
1.6		appropriately renders all information for different viewports	✓	
1.7		additional comments, glitches, crashes discovered		Too lax requirements
2	Login			
2.1		normal login process with accepted values works fine	✓	
2.2		Login button navigates to home page with accepted values	✓	
2.3		trying fields blank is rejected with a field specific helper message	✗	rejected

```
test('redirects to login page when server returns 401', async () => {
  // Arrange
  Cookies.get.mockReturnValue('mock-jwt-token');
  global.fetch = jest.fn().mockResolvedValue({
    ok: false,
    status: 401,
  });

  // Act
  render(
    <ApiContext.Provider value={mockApiContext}>
      <AuthChecker />
    </ApiContext.Provider>
  );

  // Assert
  await waitFor(() => expect(mockNavigate).toHaveBeenCalledWith('/'));
});
```

- Informal testing occurred during early development phase.
- User tests written onto a spreadsheet. Pass/fail criteria with option for comments
- Code driven tests that compartmentalize functionality of each component
- GitHub actions to assist in running tests with each commit

# Highlights and Challenges

- Comradery
- Seeing all the pieces come together and working in concert
- Trying to squeeze as much efficiency out of a single component as possible
- Discovering CSS development dependencies, bootstrap and AntD
- Component manipulation through layers of code
- Maintaining an active trello board and trying to stay on task
- Coding component tests
- Backend calls that incorporate nested data structures

END