

Front End

Login.spec.js

Home.spec.js

The screenshot shows a macOS desktop environment. In the foreground, a browser window displays a React application with a sidebar labeled "Testing" and a main area showing a list of items. The URL in the address bar is "http://localhost:3001". Below the browser is a terminal window with the following command and output:

```
zsh -c "sweng981-social-media-login git:(main) ✘ $ jest --watch
```

The terminal output shows Jest watching for changes in the "src" directory and running tests for "Login.spec.js" and "Home.spec.js". The "Login.spec.js" test includes a screenshot of the browser showing a "Sign Up" button.

In the background, the Dock shows various application icons, and the status bar at the bottom right indicates the date and time as "Sat Jun 14 11:58:00".

The screenshot shows a macOS desktop environment with several open windows. On the left, there's a sidebar with icons for Home, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help, and a user icon. The main area has a terminal window at the bottom containing Node.js code for a social media login application. Above it is a browser window displaying a simple React-based login form with fields for 'Email' and 'Password'. The browser's address bar shows the URL <http://localhost:3001/login>. The top right corner of the screen shows the date and time as Sun Jun 14 11:55:07.

```
Writing objects: 100% (89/89), 127,391 bytes | 3.64 MiB/s, done.
remote: Total 89 (delta 70), reused 0 (delta 0), pack-reused 11 local objects.
To https://github.com/mesw98/sweng801-social-media-login.git
  1f733c3..1f733c3 [new branch]  week2-social-media-login
branch 'week2-social-media-login' set up to track 'origin/week2-social-media-login'.
Switched to a new branch 'week2-social-media-login'
nmesw98@nmesw98-MacBook-Air-17: ~$ sweng801-social-media-login %
```

NavBar.spec.js

The screenshot shows a Mac OS X desktop environment. In the foreground, a web browser displays a React application with a navigation bar and some content. In the background, a Visual Studio Code (VS Code) window is open, showing the file `src/sweng601-social-media-login/index.js`. The code is a component that handles user login logic, including handling form submission and navigating to different pages based on the user's role. The VS Code interface includes tabs for other files like `package.json`, `index.html`, and `LogIn.spec.js`. The bottom of the screen features the Dock with various application icons.

The screenshot shows a Mac desktop environment with several open windows:

- Terminal:** The terminal window displays the command "git status" followed by the output: "On branch master".
- Browser:** A browser window is open to a login page for "SWENG601".
- Code Editor:** An IDE window (likely VS Code) is open to a file named "NavBar.spec.js". The code in the file is a Jest test for a navigation bar component, specifically testing the "NeedHelp" callback.
- System Dock:** The dock at the bottom of the screen contains icons for various applications including Finder, Mail, Safari, and others.

NewShow.spec.js

```
it('submit() makes error toast if api call fails', async () => {
    await wrapper.msimulate();
    await wrapper.setInitialProps();
    await wrapper.setProps({ title: 'Test Title' });
    await wrapper.vm.submit();
    expect(wrapper.state().toast).toEqual({
        type: 'error',
        message: 'Error'
    });
    expect(wrapper.state().error).toEqual(true);
    expect(wrapper.state().loading).toEqual(false);
    expect(wrapper.state().success).toEqual(false);
    expect(wrapper.state().toast).toEqual({
        type: 'error',
        message: 'Error'
    });
});
```

The screenshot shows a macOS desktop environment. In the center is a web browser window titled "sweng861-social-media-login". Below the browser is a Visual Studio Code (VS Code) interface. The left sidebar of VS Code shows a project structure for "Testing" with files like "index.html", "Card.spec.js", "NavBar.spec.js", and "NewShow.spec.js". The main code editor area contains a Jest test for the "submit" function. The test checks if an error toast is created when the API call fails. The status bar at the bottom of VS Code indicates "zsh - sweng861-social-media-login" and "ln 175, Col 3". The bottom dock of the desktop shows various application icons.

UpdateShow.spec.js

The screenshot shows a browser-based IDE interface with the following details:

- File Structure:** The left sidebar shows a tree view of the project structure under "Testing".
 - Week1: swing801-social-media-login
 - Week2: swing801-social-media-login-clone
 - Week3: swing801-social-media-login
- Code Editor:** The main area displays the file `UpdateShow.spec.js`. The code is a Jest test for an "UpdateShow" component, using a wrapper from `shallowMount`. It includes assertions for error handling and success cases.
- Terminal:** At the bottom, the terminal shows the command `zsh - swing801-social-media-login +`.
- Bottom Bar:** Includes icons for file operations like Open, Save, Find, and Run, as well as tabs for "Live Share" and "Outline".

The screenshot shows a Mac OS X desktop environment. At the top is the Dock with various application icons. Below the Dock is a toolbar with icons for Home, Back, Forward, Stop, Refresh, and Search. The main area features a terminal window, a browser window displaying a login page for 'swing861-social-media', and a code editor window for a project named 'swing861'. The code editor has tabs for 'Login.spec.js', 'Home.spec.js', 'NavBar.spec.js', 'NowShow.spec.js', 'UpdateShow.spec.js', and 'Week2.js'. The 'UpdateShow.spec.js' tab is currently active, showing Jest test code for an 'UpdateShow' component. The code includes imports for 'shallowMount', 'mount', 'expect', 'axios', 'toastr', and 'bootstrap'. It uses Jest's 'mock' function to mock 'axios' and 'toastr', and 'shallowMount' or 'mount' to render the component. The 'Login.spec.js' tab is also visible in the code editor.

DeleteShow.spec.js

The screenshot shows a macOS desktop environment. In the center is a web browser window titled "SWENG861" displaying a login page. The URL bar shows "http://localhost:3001". The browser's developer tools are open, showing the "Network" tab with several requests listed. Below the browser is a terminal window with the following Node.js code:

```
const express = require('express');
const app = express();
const port = 3001;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

At the bottom of the screen, there is a dock with various application icons, including Finder, Mail, Safari, and others. The status bar at the very bottom shows the date and time as "Sat Jun 14 12:02:39" and the battery level as "14%".

```
it('deletes modal makes error toast if api call fails', async () => {
    await wrapper.setdata('DeleteShow');
    return (
        data: [
            {
                error: 'test'
            }
        ]
    )
}

await wrapper.setdata({
    title: 'test title'
})

expect(wrapper.vm.jobTitle).toBe('DeleteShow');
expect(wrapper.vm.error).not.toBeCalled();
expect(wrapper.vm.error).not.toBeCalled();
expect(wrapper.vm.success).not.toBeCalled();
expect(wrapper.vm.success).not.toBeCalled();

it('submits makes success toast and closes modal if api call succeeds', async () => {
    await wrapper.setdata('DeleteShow');
    return (
        data: [
            {
                error: ''
            }
        ]
    )
}

Writing: objects: 399 (39), 127,29 Kuz | 820 Millz, done
Total: 39 (delta 20), reused 0 (delta 0), pack-delta: 8 from 0
removing: 127,29 Kuz | 820 Millz, done
To https://github.com/username/react-social-media-login.git
@2.0.0-alpha.1 #202111 master -> master
nscwdl@MacBook-Pro-14:~/Desktop/sweng861-social-media-login %
```

CreateAccount.spec.js

The screenshot shows a macOS desktop environment with several windows open:

- Browser:** A Safari window displaying a login form for "sweng018-social-media-login". The URL is `http://127.0.0.1:3001`. The page includes fields for "Email" and "Password", and buttons for "Log In" and "Forgot Password".
- Terminal:** A terminal window titled "sweng018-social-media-login" showing command-line output related to file uploads and database operations.
- Code Editor:** An Xcode project titled "Testing". The "EXPLORER" sidebar shows the project structure, including files like `index.html`, `App.js`, `Card.js`, `CreateAccount.spec.js` (which is currently selected), and `NavBar.js`.
- System Dock:** The Mac OS Dock at the bottom of the screen contains icons for various applications like Finder, Mail, and Safari.

The screenshot shows a macOS desktop environment with several open windows. The most prominent is a terminal window titled 'sweng81-social-media-login' which displays Jest test results:

```
Writing data to file "/Users/lukejones/Desktop/sweng81-social-media-login/test/_testResults.json"
Total: 13 tests, 13 passed (100%), 0 failing, 0 skipped
Time: 29 ms
  Run completed in 1.34 Milis, done.
  Remote: Resolving (100% (28/28)) completed with 13 local objects.
  Remote: 2026112 masters > master
  remote:shuttle@lukejones-MacBook-Pro:~/sweng81-social-media-login$
```

Below the terminal is a browser window showing a 'Login' page with fields for 'Email' and 'Password'. The browser's address bar shows 'localhost:3001'. The browser's status bar indicates 'Sat Jun 12 14:23:36'.

Card.spec.js

A screenshot of a macOS desktop environment. The window title is "Testing [File Edit View Go Run Terminal Window Help SWENGS861]". The main content area shows an "EXPLORER" sidebar with project files like "package.json", "users.test.js", "Login.spec.js", "Home.spec.js", "NavBar.spec.js", and "Card.spec.js". A file named "Card.spec.js" is open in the editor, containing a Jest test for a "Card" component. The test uses "shallowMount" to render the component and then checks various properties and events. The status bar at the bottom shows "In 46 Col 1 Spaces 4 UTF-8 LF Babel JavaScript". The dock at the bottom has icons for Finder, Home, Mail, Safari, and others.

```
Week2 / swing861-social-media-login < Front-end > Card.spec.js > describe("Card" callback) { // if show data renders if values are true 1 import { shallowMount, mount } from 'ava/test-utils' 2 import Card from '@components/card.vue' 3 4 describe('Card') { 5   const wrapper = shallowMount(Card, { 6     propsData: { 7       show: null 8     }, 9     global: { 10       stubs: { 11         router: true 12       } 13     } 14   }) 15 16   it('show card does not render if show is falsy', () => { 17     expect(wrapper.find('.card').exists()).toBeFalsy() 18   }) 19 20   it('show card renders if show is truthy', async () => { 21     await wrapper.setProps({ show: true }) 22 23     expect(wrapper.find('.card').exists()).toBeTruthy() 24   }) 25 26   it('UpdateIcon emits updateShow event on click', async () => { 27     await wrapper.find('#bin-bin-info').trigger('click') 28 29     expect(wrapper.emitted('updateShow')).toBeTruthy() 30   }) 31 }
```

Back-End

google.test.js

The screenshot shows a Mac OS X desktop environment. At the top is a dark-themed menu bar with options like Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help, and a PLC icon. The date and time 'Sun Jun 14 12:05:26' are also at the top right.

The main window is a code editor with tabs for 'Testing' and 'Tab 1'. The 'Tab 1' tab is active, showing the file 'google-test.js'. The code in the file is related to a 'sweng681-social-media-login' project, specifically a 'back-end' test. It includes imports for 'chai', 'expect', 'request', and 'superagent'. The code uses 'describe' blocks for 'GET /auth/google' and 'GET /auth/google/callback' tests, setting up mock data and assertions for responses.

On the left side of the code editor is an 'EXPLORER' sidebar showing the project structure. It includes a 'SWENG681' folder containing 'User1', 'User2', and 'Week2', and a 'Week2' folder containing 'sweng681-social-media-login', 'back-end', and 'front-end'. Inside 'back-end', there are files like 'index.html', 'README.md', 'google-test.js', 'LinkedIn.test.js', 'shows.test.js', 'users.test.js', 'config', 'logger', 'node_modules', 'routes', 'secrets', 'env', 'logger-log.json', 'package.json', 'server.js', 'docs', 'front-end', 'vscode', 'coverage', 'dist', 'node_modules', 'public', 'src', and 'OUTLINE' and 'TIMELINE' sections.

At the bottom of the screen is a dock with various application icons: Finder, Mail, Safari, Calendar, Reminders, Stocks, Wallet, News, TV, App Store, iBooks, iTunes Store, iBooks Author, iMovie, Final Cut Pro X, GarageBand, iPhoto, Preview, System Preferences, and Terminal. A terminal window is open at the bottom right, showing command-line output related to the Java application.

```
git:(master) ✘ node index.js
Writing objects: 100% (39/39), 127.29 KiB | 3.64 MiB/s
Total 39 (delta 28), reused 0 delta(s), pack-reused 0 (delta(s))
remote: Resolving deltas: 100% (28/28) done.
To https://github.com/nathanhs/google-social-media-login.git
 * [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
You can now pull or push to this branch.
nicheshar@Nathanhs-MacBook-Air:~$ node google-test.js
```

```
const request = require('superagent');
const config = require('../config/db.json');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const axios = require('axios');

jest.mock('bcryptjs', () => {
  return {
    hash: jest.fn(),
    compare: jest.fn()
  }
});

jest.mock('../jsonwebtoken', () => {
  return {
    sign: jest.fn()
  }
});

jest.mock('../..../config/db.json', () => {
  return {
    query: jest.fn(),
    on: jest.fn()
  }
});

jest.mock('axios', () => {
  return {
    get: jest.fn(),
    post: jest.fn()
  }
});

describe('Google', () => {
  it('should log in user', () => {
    const response = {
      data: {
        id: '1234567890',
        name: 'Nathan Hsieh',
        email: 'nathanhsieh@gmail.com'
      }
    };
    const token = jwt.sign({ id: '1234567890' }, config.secret);
    const result = Google.googleLogin(response, token);
    expect(result).toEqual({
      id: '1234567890',
      name: 'Nathan Hsieh',
      email: 'nathanhsieh@gmail.com'
    });
  });
});
```

linkedin.test.js

The screenshot shows a macOS desktop environment with several open windows. The most prominent is a terminal window titled 'SWEN9801' which contains the following Node.js code:

```
const express = require('express');
const axios = require('axios');
const cookieParser = require('cookie-parser');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

const app = express();
app.use(cookieParser());
app.use(express.json());

const users = [
    {
        id: 1,
        name: 'John Doe',
        email: 'john.doe@example.com',
        password: bcrypt.hashSync('password123', 8),
        tokens: []
    }
];

function generateToken(id) {
    return jwt.sign({ id }, 'secretKey');
}

app.get('/api/login', (req, res) => {
    const { email, password } = req.query;
    const user = users.find(u => u.email === email);
    if (!user || !bcrypt.compareSync(password, user.password)) {
        return res.status(401).json({ error: 'Invalid credentials' });
    }
    const token = generateToken(user.id);
    res.cookie('token', token).json({ user, token });
});

app.get('/api/logout', (req, res) => {
    const token = req.cookies.token;
    if (!token) {
        return res.status(401).json({ error: 'User not logged in' });
    }
    const user = users.find(u => u.tokens.includes(token));
    if (!user) {
        return res.status(401).json({ error: 'User not found' });
    }
    user.tokens = user.tokens.filter(t => t !== token);
    res.clearCookie('token').json({ message: 'Logout successful' });
});

app.get('/api/me', (req, res) => {
    const token = req.cookies.token;
    if (!token) {
        return res.status(401).json({ error: 'User not logged in' });
    }
    const user = users.find(u => u.tokens.includes(token));
    if (!user) {
        return res.status(401).json({ error: 'User not found' });
    }
    res.json({ user });
});

app.get('/api/messages', (req, res) => {
    const token = req.cookies.token;
    if (!token) {
        return res.status(401).json({ error: 'User not logged in' });
    }
    const user = users.find(u => u.tokens.includes(token));
    if (!user) {
        return res.status(401).json({ error: 'User not found' });
    }
    const messages = [
        {
            id: 1,
            user_id: 1,
            content: 'Hello John!',
            timestamp: '2023-10-01T12:00:00Z'
        },
        {
            id: 2,
            user_id: 1,
            content: 'How are you?',
            timestamp: '2023-10-01T12:15:00Z'
        },
        {
            id: 3,
            user_id: 1,
            content: 'I am good, thanks for asking!',
            timestamp: '2023-10-01T12:30:00Z'
        }
    ];
    res.json({ messages });
});

app.listen(3001, () => console.log('Server listening on port 3001'));
```

The terminal window is part of a larger interface with multiple tabs and panes. The top navigation bar includes 'File', 'Edit', 'Selection', 'View', 'Do', 'Run', 'Terminal', 'Window', and 'Help'. A status bar at the bottom shows system information like 'Lan 16', 'Cal 24', 'Topics 4', 'UTF-8', 'LF', and 'JavaScript'. The bottom dock contains icons for various applications including Finder, Mail, Safari, and the terminal.

The screenshot shows a macOS desktop environment with several open windows. The terminal window at the bottom contains Node.js code for a social media login application, specifically for LinkedIn. The browser window shows a login page for 'LinkedIn' with fields for 'Email or phone number' and 'Password'. The top of the screen features the macOS Dock with various application icons.

```
const express = require('express');
const axios = require('axios');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const config = require('../config/db.js');

const app = express();
app.use(express.json());
app.use('/api/auth', authRoutes);

// Routes
app.get('/api/auth/test', (req, res) => {
  res.send('Auth API is working');
});

// User routes
app.get('/api/auth/login', (req, res) => {
  const { email, password } = req.query;
  const user = {
    id: 1,
    name: 'John Doe',
    email: 'john.doe@example.com',
    password: 'password123'
  };

  if (user.email === email && user.password === password) {
    const token = jwt.sign({ user }, config.secret);
    res.json({ token });
  } else {
    res.status(401).json({ message: 'Invalid credentials' });
  }
});

app.post('/api/auth/register', (req, res) => {
  const { name, email, password } = req.body;
  const user = {
    id: 1,
    name: 'John Doe',
    email: 'john.doe@example.com',
    password: 'password123'
  };

  if (!user) {
    res.status(400).json({ message: 'User already exists' });
  } else {
    res.json({ message: 'User registered successfully' });
  }
});

// Admin routes
app.get('/api/admin/test', (req, res) => {
  res.send('Admin API is working');
});

// Error handling
app.use((err, req, res, next) => {
  console.error(err);
  res.status(500).json({ message: 'Internal server error' });
});

module.exports = app;
```

shows.test.js

```
Code File Edit Selection View Go Run Terminal Window Help
SWENG801
shows.test.js
Week2 > sweng801-social-media-login > back-end > __tests__ > shows.test.js
describe('DELETE /deleteShow/:id', () => {
  const pool = require('../config/db.js');
  const server = require('../server');
  const jwt = require('jsonwebtoken');

  let token;
  let showId;
  let res;
  let db;

  beforeAll(() => {
    pool.query('TRUNCATE TABLE shows');
  });

  it('createsShow returns error message if validation fails', async () => {
    const res = await request(server).post('/createShow').send({
      title: 'Show 1',
      description: 'This is a test show'
    });
    expect(res.body.title).toStrictEqual('error', 'Validation failed');
    expect(pool.query).not.toBeCalled();
  });

  it('createsShow makes correct insert query to database', async () => {
    const res = await request(server).post('/createShow').send({
      title: 'Show 1',
      description: 'This is a test show'
    });
    expect(res.body.title).toStrictEqual('Show 1');
  });

  const res = await request(server).post('/createShow').send({
    title: 'Show 1',
    description: 'This is a test show'
  });
  expect(res.status).toBe(201);
  expect(res.body.title).toStrictEqual('Show 1');
});

it('getShows returns created show', () => {
  const res = await request(server).get('/getShows');
  expect(res.body.shows[0].title).toStrictEqual('Show 1');
});
```

```
Code File Edit Selection View Go Run Terminal Window Help
SWENG801
shows.test.js
Week2 > sweng801-social-media-login > back-end > __tests__ > shows.test.js
describe('DELETE /deleteShow/:id', () => {
  const pool = require('../config/db.js');
  const server = require('../server');
  const jwt = require('jsonwebtoken');

  let token;
  let showId;
  let res;
  let db;

  beforeAll(() => {
    pool.query('TRUNCATE TABLE shows');
  });

  it('createsShow returns error message if validation fails', async () => {
    const res = await request(server).post('/createShow').send({
      title: 'Show 1',
      description: 'This is a test show'
    });
    expect(res.body.title).toStrictEqual('error', 'Validation failed');
    expect(pool.query).not.toBeCalled();
  });

  it('createsShow makes correct insert query to database', async () => {
    const res = await request(server).post('/createShow').send({
      title: 'Show 1',
      description: 'This is a test show'
    });
    expect(res.body.title).toStrictEqual('Show 1');
  });

  const res = await request(server).post('/createShow').send({
    title: 'Show 1',
    description: 'This is a test show'
  });
  expect(res.status).toBe(201);
  expect(res.body.title).toStrictEqual('Show 1');
});

it('getShows returns created show', () => {
  const res = await request(server).get('/getShows');
  expect(res.body.shows[0].title).toStrictEqual('Show 1');
});
```

users.test.js

```
Code File Edit Selection View Go Run Terminal Window Help
SWENG801
users.test.js
Week2 > sweng801-social-media-login > back-end > __tests__ > users.test.js
describe('POST /createAccount', () => {
  const pool = require('../config/db.js');
  const server = require('../server');
  const jwt = require('jsonwebtoken');

  let token;
  let user;
  let res;
  let db;

  beforeAll(() => {
    pool.query('TRUNCATE TABLE users');
  });

  it('createsAccount returns error message if user already exists', async () => {
    const res = await request(server).post('/createAccount').send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'password123'
    });
    expect(res.body.message).toStrictEqual('User with this email already exists');
    expect(pool.query).not.toBeCalled();
  });

  it('createsAccount makes correct insert query to database', async () => {
    const res = await request(server).post('/createAccount').send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'password123'
    });
    expect(res.status).toBe(201);
    expect(res.body.message).toStrictEqual('User created successfully');
  });

  const res = await request(server).post('/createAccount').send({
    name: 'Test User',
    email: 'test@example.com',
    password: 'password123'
  });
  expect(res.status).toBe(201);
  expect(res.body.message).toStrictEqual('User created successfully');
});
```

```
Code File Edit Selection View Go Run Terminal Window Help
SWENG801
users.test.js
Week2 > sweng801-social-media-login > back-end > __tests__ > users.test.js
describe('POST /createAccount', () => {
  const pool = require('../config/db.js');
  const server = require('../server');
  const jwt = require('jsonwebtoken');

  let token;
  let user;
  let res;
  let db;

  beforeAll(() => {
    pool.query('TRUNCATE TABLE users');
  });

  it('createsAccount returns error message if user already exists', async () => {
    const res = await request(server).post('/createAccount').send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'password123'
    });
    expect(res.body.message).toStrictEqual('User with this email already exists');
    expect(pool.query).not.toBeCalled();
  });

  it('createsAccount should make two database queries if user does not exist', async () => {
    // Mock first query returns no user
    pool.query.mockImplementationOnce(() => ({ rows: [] }));
    const res = await request(server).post('/createAccount').send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'password123'
    });
    expect(res.status).toBe(201);
    expect(res.body.message).toStrictEqual('User created successfully');
  });

  const res = await request(server).post('/createAccount').send({
    name: 'Test User',
    email: 'test@example.com',
    password: 'password123'
  });
  expect(res.status).toBe(201);
  expect(res.body.message).toStrictEqual('User created successfully');
});
```