

```

1  //Backstory.js
2  // Written by Julia Workum and Lia Ferguson
3  class Backstory extends React.Component {
4      constructor(props) {
5          super(props);
6      }
7
8      render() {
9          return (
10             <Container fluid="md">
11                 <h2 className='sub-headers' >Backstory</h2>
12                 <p>
13                     Your best friend and boss of 5 years, Tony Stark, has just
14                     been found dead in the office break room right next to
15                     the coffee machine.
16                     Unfortunately, the police don't have any suspects and the
17                     case has gone cold.
18                     However, you think that one of your coworkers must be
19                     responsible.
20                     In order to see who's had access to the breakroom, you
21                     know you'll have to break into the secret employee
22                     database on Tony's computer.
23                     Now, it's up to you to use your SQL skills and find out
24                     who murdered Tony Stark.
25                 </p>
26                 <Row className="justify-content-md-center">
27                     <Col>
28                         <Button variant="outline-primary float-right" href="/
29                         rules">Next</Button>
30                     </Col>
31                 </Row>
32             </Container>
33         );
34     }
35 }
36 export default Backstory;
37
38 //confirmsuspect.js
39 //Lia Ferguson
40 class ConfirmSuspect extends React.Component {
41     constructor(props) {
42         super(props);
43
44         this.state = { isClicked: false, isCorrect: false };
45         this.handleClickSuspectButton = this.handleClickSuspectButton.bind
46             (this);
47     }
48 }

```

```

41   handleCorrectSuspectButton(e) {
42       this.setState({ isClicked: true });
43       if (e.target.value == 'correct') {
44           this.setState({ isCorrect: true });
45           this.props.suspectConfirmCorrect(true);
46       } else {
47           this.setState({ isCorrect: false });
48           this.props.suspectConfirmCorrect(false);
49       }
50   }
51
52   render() {
53       let buttonResponse = null;
54       if (this.state.isClicked) {
55           if (this.state.isCorrect) {
56               buttonResponse = <div><p className='helper-text'>You're right, ↗
                    both Natasha and Bruce didn't check into the building until ↗
                    after Tony was found in the break room, so
57               it is unlikely that either of them are responsible for Tony's ↗
                    death.
58               </p></div>
59           } else {
60               buttonResponse = <div className='center-text'><p ↗
                    className='helper-text'>Are you sure? Remember, Tony ↗
                    accessed the building at <b>11:30am</b> and was found dead ↗
                    <b>before noon</b></p></div>;
61           }
62       }
63
64       return (
65           <Container>
66               <div>
67                   <h5>Re-evaluate Suspect List</h5>
68                   <p className='helper-text'>Based on the results of your ↗
                    query, do you think that Natasha and Bruce should still be ↗
                    on our suspect list?</p>
69               </div>
70
71               <Row>
72                   <div className='button-group-suspect'>
73                       <ButtonGroup>
74                           <Button variant='primary' value='incorrect' ↗
                    onClick={this.handleCorrectSuspectButton}>Yes</Button>
75                           <Button variant='primary' value='correct' onClick= ↗
                    {this.handleCorrectSuspectButton}>No</Button>
76                       </ButtonGroup>
77                   </div>
78                   {buttonResponse}
79               </Row>

```

```

80     </Container>
81   );
82 }
83 }
84
85 export default ConfirmSuspect;
86
87 //Header.js
88 //Lia Ferguson and Julia Workum
89 class Header extends React.Component {
90   constructor(props) {
91     super(props);
92   }
93
94   render() {
95     return (
96       <Container fluid='true'>
97         <Jumbotron>
98           <h1 class="display-4 header-content">Welcome to SQL Murder
99             Mystery!</h1>
100           <p class="lead header-content">Can you figure out who
101             murdered Tony Stark?</p>
102         </Jumbotron>
103       </Container>
104     );
105   }
106 }
107
108 export default Header;
109
110 //hint.js
111 // Written by Lia Ferguson
112
113 class Hint extends React.Component {
114   constructor(props) {
115     super(props);
116   }
117
118   render() {
119     return (
120       <Container>
121         <Row className="justify-content-md-center">
122           <Col xs={8}>
123             <Accordion className='hint'>
124               <Card>
125                 <Accordion.Toggle as={Card.Header}
126                   eventKey="0">
127                   Hint
128                 </Accordion.Toggle>

```

```
126         <Accordion.Collapse eventKey="0">
127             <Card.Body>{this.props.hint}</Card.Body>
128         </Accordion.Collapse>
129     </Card>
130 </Accordion>
131 </Col>
132 </Row>
133 </Container>
134 );
135 }
136 }
137
138 export default Hint;
139
140 //loginsql.js
141 // lines 40, 48-50 written by Andrew Fecher
142 // all other code written by Lia Ferguson
143 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
144
145 class LoginSQL extends React.Component {
146     constructor(props) {
147         super(props);
148
149         this.state = { isClicked: false, isQuerySuccessful: false, user_id:
150             '', password: '', results: '', correctResults: false, errorMessage:
151             '' };
152
153         this.handleQuery = this.handleQuery.bind(this);
154         this.handleUserIdChange = this.handleUserIdChange.bind(this);
155         this.handlePasswordChange = this.handlePasswordChange.bind(this);
156     }
157
158     handleUserIdChange(e) {
159         e.preventDefault()
160         this.setState({ user_id: e.target.value });
161     }
162
163     handlePasswordChange(e) {
164         e.preventDefault()
165         this.setState({ password: e.target.value });
166     }
167
168     async handleQuery() {
169         var response = await this.executeQuery(this.state.user_id,
170             this.state.password);
171         var isQuerySuccessful = response.isQuerySuccessful == 'true' ? true :
172             false;
173         var correctResults = response.correctResults == 'true' ? true : false;
174         this.setState({ isQuerySuccessful: isQuerySuccessful });
175     }
176 }
```

```
171     this.setState({ correctResults: correctResults });
172     this.setState({ isClicked: true });
173     this.setState({ errorMessage: response.error });
174     if (isQuerySuccessful && correctResults) {
175         this.props.batchSqlCorrect(true);
176     } else {
177         this.props.batchSqlCorrect(false);
178     }
179     if (this.props.processResults !== null) {
180         this.props.processResults(response.results);
181     }
182 }
183
184 async executeQuery(user_id, pwd, isQuerySuccessful) {
185     const response = await fetch(BACKEND_API_URL + "/login_query", {
186         method: "POST",
187         mode: 'cors',
188         headers: {
189             'Content-Type': 'application/json'
190         },
191         body: JSON.stringify({
192             isQuerySuccessful: isQuerySuccessful,
193             user_id: user_id,
194             password: pwd,
195             game_step: this.props.game_step
196         })
197     })
198     return await response.json();
199 }
200
201 render() {
202     let queryResponse, continueButton;
203     if (this.state.isClicked && this.state.isQuerySuccessful &&  ↗
204         this.state.correctResults) {
205         queryResponse = <div className="instruction-div">
206             <p className="helper-text">
207                 {this.props.congratsMessage}
208             </p>
209             </div>;
210         continueButton = <Button variant="outline-primary float-right"  ↗
211             href="/step2">Continue</Button>;
212     } else if (this.state.isClicked && (!this.state.isQuerySuccessful || !  ↗
213         this.state.correctResults)) {
214         queryResponse = <div className="instruction-div">
215             <p>
216                 {this.state.errorMessage}
217             </p>
218             </div>;
219         continueButton = null;
```

```

217     }
218     return (
219         <Container>
220             <Row className="justify-content-md-center">
221                 <Col xs={8}>
222                     <Form>
223                         <div align='center' className='login-form'>
224                             <h3 className='sub-headers'>Login</h3>
225                             <Form.Group controlId='username'>
226                                 <Form.Label className='login-  ↗
227                                     labels'>User_ID</Form.Label>
228                                 <Form.Control value={this.state.user_id}  ↗
229                                     type='username' placeholder="Enter User_ID here" onChange=  ↗
230                                     {this.handleUserIdChange}></Form.Control>
231                                 </Form.Group>
232                                 <Form.Group controlId='password'>
233                                     <Form.Label className='login-labels'  ↗
234                                     >Password</Form.Label>
235                                     <Form.Control value={this.state.password}  ↗
236                                     type='password' placeholder="Enter password here"  ↗
237                                     onChange={this.handlePasswordChange} ></Form.Control>
238                                 </Form.Group>
239                                 <Button className='login-button'  ↗
240                                     variant='primary' onClick={this.handleQuery}>Login</  ↗
241                                     Button>
242                                     {queryResponse}
243                                 </div>
244                             </Form>
245                         </Col>
246                     </Row>
247                 </Container>
248             );
249     }
250 }
251
252 export default LoginSQL;
253
254 //paddedcell.js
255 // Written by Lia Ferguson
256 class PaddedCell extends React.Component {
257     constructor(props) {
258         super(props);
259
260         this.state = { buttonClicked: false, correctQQ1: false, correctQQ2:  ↗
261             false, correctQQ3: false };
262         this.handleClick = this.handleClick.bind(this);
263         this.handleCorrectChoice = this.handleCorrectChoice.bind(this);
264     }
265 }

```



```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 8
295 qq1Responses['C'] = 'Trojan horses use social engineering or some  ↗
    other masking technique to get users to download malicious software  ↗
    under the guise of legitimate software. In this case, you thought  ↗
    you were downloading a helpful text file which turned out to be a  ↗
    false confession.';
296 qq1Responses['D'] = 'Polymorphic viruses change themselves throughout  ↗
    the attack in order to mask their signatures.';
297
298 const qq2Question = 'The shortcut popup window was a simplified and  ↗
    obvious example of a technique called _____ where applications  ↗
    might try to lure attackers in with desirable information in order  ↗
    to get more information about the attacker.';

299
300 const qq2Answers = [
301     <Card.Text className="answer-text"> Baiting <br></br><br></br></  ↗
        Card.Text>,
302     <Card.Text className="answer-text"> Trap and Trace <br></br><br></  ↗
        br></Card.Text>,
303     <Card.Text className="answer-text"> Honey Netting <br></br><br></  ↗
        br></Card.Text>,
304     <Card.Text className="answer-text"> Being Sneaky <br></br><br></  ↗
        br></Card.Text>
305 ];
306
307 const qq2Header = new Map();
308 qq2Header['A'] = 'Incorrect';
309 qq2Header['B'] = 'Correct';
310 qq2Header['C'] = 'Incorrect';
311 qq2Header['D'] = 'Incorrect';
312
313 const qq3Question = 'Because the game tried to pin the murder of Tony  ↗
    Stark on you after you fell for the shortcut padded cell which is  ↗
    all kinds of illegal, this trap and trace scenario fits most closely  ↗
    under the category of...';

314
315 const qq3Answers = [
316     <Card.Text className="answer-text"> Enticement <br></br><br></  ↗
        br></Card.Text>,
317     <Card.Text className="answer-text"> Entrapment <br></br><br></  ↗
        br></Card.Text>
318 ];
319
320 const qq3Header = new Map();
321 qq3Header['A'] = 'Incorrect';
322 qq3Header['B'] = 'Correct';
323
324 const qq3Responses = new Map();
325 qq3Responses['A'] = 'Enticement involves dangling desirable pieces of  ↗
    information in order to attract attackers to a specific part of a  ↗

```





```

367         {qq1Responses[ 'B' ]}
368         </Popover.Content>
369     </Popover>
370 }>
371     <Button className="question-button"
variant="outline-primary" size="sm" > B </Button>
372     </OverlayTrigger>
373     <Card.Body className='question-text'>{qq1Answers
[1]}</Card.Body>
374     </Card>
375 </CardGroup>
376 </Row>
377 <Row className="justify-content-md-center">
378     <CardGroup>
379         <Card style={{ width: '18rem' }} bg="light">
380             <OverlayTrigger
381                 trigger="click"
382                 key='bottom'
383                 rootClose
384                 placement='bottom'
385                 overlay={
386                     <Popover id='popover-positioned-bottom'>
387                         <Popover.Title as="h3">{qq1Header
['C']}</Popover.Title>
388                         <Popover.Content>
389                             {qq1Responses[ 'C' ]}
390                         </Popover.Content>
391                     </Popover>
392                 }>
393                 <Button className="question-button"
variant="outline-primary" size="sm" value="qq1" onClick=
{this.handleCorrectChoice}> C </Button>
394             </OverlayTrigger>
395             <Card.Body className='question-text'>{qq1Answers
[2]}</Card.Body>
396             </Card>
397             <Card style={{ width: '18rem' }} bg="light">
398                 <OverlayTrigger
399                     trigger="click"
400                     key='bottom'
401                     rootClose
402                     placement='bottom'
403                     overlay={
404                         <Popover id='popover-positioned-bottom'>
405                             <Popover.Title as="h3">{qq1Header
['D']}</Popover.Title>
406                             <Popover.Content>
407                                 {qq1Responses[ 'D' ]}
408                             </Popover.Content>

```

```

409         </Popover>
410     }>
411     <Button className="question-button"
variant="outline-primary" size="sm"> D </Button>
412     </OverlayTrigger>
413     <Card.Body className='question-text'>{qq1Answers
[3]}</Card.Body>
414     </Card>
415 </CardGroup>
416 </Row>
417 </Container>;
418
419 question2 = <Container>
420   <Row className="justify-content-md-center">
421     <p className="q-text">
422       <b>Question</b><br></br>
423       {qq2Question}
424     </p>
425   </Row>
426   <Row className="justify-content-md-center">
427     <CardGroup>
428       <Card style={{ width: '18rem' }} bg="light">
429         <OverlayTrigger
430           trigger="click"
431           key='top'
432           rootClose
433           placement='top'
434           overlay={
435             <Popover id='popover-positioned-top'>
436               <Popover.Title as="h3">{qq2Header
['A']}</Popover.Title>
437             </Popover>
438           }>
439             <Button className="question-button"
variant="outline-primary" size="sm"> A </Button>
440             </OverlayTrigger>
441             <Card.Body className='question-text'>{qq2Answers
[0]}</Card.Body>
442             </Card>
443             <Card style={{ width: '18rem' }} bg="light">
444               <OverlayTrigger
445                 trigger="click"
446                 key='top'
447                 rootClose
448                 placement='top'
449                 overlay={
450                   <Popover id='popover-positioned-top'>
451                     <Popover.Title as="h3">{qq2Header
['B']}</Popover.Title>

```

```

452         </Popover>
453     }>
454     <Button className="question-button"
variant="outline-primary" size="sm" value="qq2" onClick=
{this.handleCorrectChoice}> B </Button>
455     </OverlayTrigger>
456     <Card.Body className='question-text'>{qq2Answers
[1]}</Card.Body>
457 </Card>
458 </CardGroup>
459 </Row>
460 <Row className="justify-content-md-center">
461     <CardGroup>
462         <Card style={{ width: '18rem' }} bg="light">
463             <OverlayTrigger
464                 trigger="click"
465                 key='bottom'
466                 rootClose
467                 placement='bottom'
468                 overlay={
469                     <Popover id='popover-positioned-bottom'>
470                         <Popover.Title as="h3">{qq2Header
['C']}</Popover.Title>
471                     </Popover>
472                 }>
473                 <Button className="question-button"
variant="outline-primary" size="sm"> C </Button>
474                 </OverlayTrigger>
475                 <Card.Body className='question-text'>{qq2Answers
[2]}</Card.Body>
476             </Card>
477             <Card style={{ width: '18rem' }} bg="light">
478                 <OverlayTrigger
479                     trigger="click"
480                     key='bottom'
481                     rootClose
482                     placement='bottom'
483                     overlay={
484                         <Popover id='popover-positioned-bottom'>
485                             <Popover.Title as="h3">{qq2Header
['D']}</Popover.Title>
486                         </Popover>
487                     }>
488                     <Button className="question-button"
variant="outline-primary" size="sm"> D </Button>
489                     </OverlayTrigger>
490                     <Card.Body className='question-text'>{qq2Answers
[3]}</Card.Body>
491                 </Card>

```

```

492         </CardGroup>
493     </Row>
494 </Container>;
495
496 question3 = <Container>
497     <Row className="justify-content-md-center">
498         <p className="q-text">
499             <b>Question</b><br></br>
500             {qq3Question}
501         </p>
502     </Row>
503     <Row className="justify-content-md-center">
504         <CardGroup>
505             <Card style={{ width: '18rem' }} bg="light">
506                 <OverlayTrigger
507                     trigger="click"
508                     key='top'
509                     rootClose
510                     placement='top'
511                     overlay={
512                         <Popover id='popover-positioned-top'>
513                             <Popover.Title as="h3">{qq3Header
514                                 [ 'A' ]}</Popover.Title>
515                             <Popover.Content>
516                                 {qq3Responses[ 'A' ]}
517                             </Popover.Content>
518                         </Popover>
519                     }>
520                     <Button className="question-button"
521                         variant="outline-primary" size="sm"> A </Button>
522                 </OverlayTrigger>
523                 <Card.Body className='question-text'> {qq3Answers
524                     [0]}</Card.Body>
525             </Card>
526             <Card style={{ width: '18rem' }} bg="light">
527                 <OverlayTrigger
528                     trigger="click"
529                     key='top'
530                     rootClose
531                     placement='top'
532                     overlay={
533                         <Popover id='popover-positioned-top'>
534                             <Popover.Title as="h3">{qq3Header
535                                 [ 'B' ]}</Popover.Title>
536                             <Popover.Content>
537                                 {qq3Responses[ 'B' ]}
538                             </Popover.Content>
539                         </Popover>
540                     }>

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 14
537         <Button className="question-button"  ↗
            variant="outline-primary" size="sm" value="qq3" onClick=  ↗
            {this.handleCorrectChoice}> B </Button>
538         </OverlayTrigger>
539         <Card.Body className='question-text'>{qq3Answers  ↗
            [1]}</Card.Body>
540         </Card>
541     </CardGroup>
542 </Row>
543 </Container>;
544 }
545
546 let sendoffMessage, exitButton = null;
547 if (this.state.correctQQ1 && this.state.correctQQ2 &&  ↗
    this.state.correctQQ3) {
548     sendoffMessage = <p className="helper-text">Congratulations,  ↗
        you've found your way out of the padded cell! Once you find out  ↗
        who the real murder is, the fake confession will be removed from  ↗
        your computer.</p>
549     exitButton = <div className="pad-cell-button"><Button  ↗
        variant='primary' href='/step5'>Exit Padded Cell</Button></div>;
550 }
551 return (
552     <Container>
553         <h2 className='sub-headers'>PADDED CELL</h2>
554         <h5>You've Been Framed!!</h5>
555         <p>
556             Oh no! You fell for a padded cell! It seems like the  ↗
            murderer is attempting to cover up their tracks.
557             They must have hacked into the environment and placed that  ↗
            shortcut offer there in attempts to slow down
558             anyone trying to retrace their steps through the database!  ↗
            <br></br><br></br>
559
560             To make matters worse, the company was just alerted that  ↗
            there was a document found on your computer titled 'For  ↗
            Police'
561             that contains a dated, full confession to Tony's murder!  ↗
            You've been framed! <br></br><br></br>
562
563             Don't panic, we can fix this. It seems like the murderer  ↗
            left a back door that used during their testing open. All  ↗
            you have to do
564             is answer the following security questions! <br></br>  ↗
            <br><br></br>
565         </p>
566         <p className="helper-text"><b>Answer the following security  ↗
            questions correctly in order to escape the padded cell and  ↗
            clear your name!</b></p>

```

```

567         <div className="begin-button">
568             <Button variant='primary' onClick={this.handleClick} >
569                 > Begin </Button>
570             </div>
571             <br></br><br></br>
572             {question1}
573             <br></br><br></br>
574             {question2}
575             <br></br><br></br>
576             {question3}
577             <br></br><br></br>
578             {sendoffMessage}
579             {exitButton}
580         </Container>
581     );
582 }
583 }
584
585 export default PaddedCell;
586
587 //practice.js
588 //Written in HTML by Thomas Chmura and reactified/edited by Andrew Fecher
589 class Practice extends React.Component {
590     constructor(props) {
591         super(props);
592         this.state = { query: '', queryResponse: '', sql1: '', sql1Response:
593             '', sql2: '', sql2Response: '' }
594
595         this.handleClick = this.handleClick.bind(this);
596         this.handleClickChange = this.handleClickChange.bind(this);
597         this.handleSQL1 = this.handleSQL1.bind(this);
598         this.handleSQL1Change = this.handleSQL1Change.bind(this);
599         this.handleSQL2 = this.handleSQL2.bind(this);
600         this.handleSQL2Change = this.handleSQL2Change.bind(this);
601     }
602     handleClickChange(e) {
603         e.preventDefault()
604         this.setState({ query: e.target.value });
605     }
606     handleClick() {
607         if (this.state.query == 'SELECT Password FROM Users WHERE name =
608             \'John Doe\'' ) {
609             this.setState({ queryResponse: 'This is an accurate query to get
610                 John Doe' });
611         }
612         else {
613             this.setState({ queryResponse: 'try again' });
614         }
615     }

```

```

612     }
613     handleSQL1() {
614         if (this.state.sql1 == '" OR 1=1 --') {
615             this.setState({ sql1Response: 'This is an accurate SQL Injection!' });
616         }
617         else {
618             this.setState({ sql1Response: 'try again' });
619         }
620     }
621     handleSQL1Change(e) {
622         e.preventDefault()
623         this.setState({ sql1: e.target.value });
624     }
625     handleSQL2() {
626         if (this.state.sql2 == '" ; SELECT Computer_Number FROM Computers WHERE
        Name = \'John Doe\' --') {
627             this.setState({ sql2Response: 'This is an accurate SQL Batch Injection!' });
628         }
629         else {
630             this.setState({ sql2Response: 'try again' });
631         }
632     }
633     handleSQL2Change(e) {
634         e.preventDefault()
635         this.setState({ sql2: e.target.value });
636     }
637     render() {
638         return (
639             <Container fluid='md'>
640                 <h2 className='sub-headers'>Let's Practice!</h2>
641                 <p>SQL Injection can occur when a user is prompted for input
        on a piece of information, such as a User ID. Instead of
        putting in their ID, they would input a SQL Statement that
        runs through the systems database retrieves sensitive
        information.
642                 Let's run through this process with a few simple practice
        problems so we can get your feet wet before we embark on the
        real challenge! </p>
643                 <br />
644                 <p>In the form below, we're going to retrieve the password
        information for "John Doe". In the "Username" slot, input
        the following SQL Query:
645                 <b>SELECT Password FROM Users WHERE name = 'John Doe' </b>
646                 </p>
647                 <Row className="justify-content-md-center">
648                     <Col xs={8}>
649                         <Form>

```



```

650         <div align='center' className='login-form'>
651             <h3 className='sub-headers'>SQL Query</h3>
652             <Form.Group controlId='username'>
653                 <Form.Label className='login-      ↗
labels'>Query</Form.Label>
654                 <Form.Control value={this.state.query}      ↗
type='username' placeholder="Enter query here" onChange=      ↗
{this.handleQueryChange}></Form.Control>
655             </Form.Group>
656             <Button className='login-button'      ↗
variant='primary' onClick={this.handleQuery}>Run Query</      ↗
Button>
657             <p> {this.state.queryResponse} </p>
658         </div>
659     </Form>
660 </Col>
661 </Row>
662 <br />
663 <p> Now lets see how we can utilize the above for SQL      ↗
Injection. For example, if we want to login as the admin, we      ↗
can use the fact that
664     the admin is generally first entry in the user database. So      ↗
all we would need to do is to get the SQL statement above      ↗
to return at least the admins information <br />
665     So, we know that the login will typically use something      ↗
like: SELECT User_ID FROM Users WHERE User_ID = "$userID"      ↗
AND Password = "$password" <br />
666     If we were to type in: " OR 1=1 -- in the user name space,      ↗
the SQL query would look like... <br />
667     SELECT User_ID FROM Users WHERE User_ID = "" OR 1=1 --"      ↗
AND Password = "$password"<br />
668     Since -- is a comment and 1=1 is true, it will always      ↗
return all users <br />
669     <b>Type in : " OR 1=1 -- below to sql inject into the      ↗
system! </b>
670 </p>
671 <Row className="justify-content-md-center">
672     <Col xs={8}>
673         <Form>
674             <div align='center' className='login-form'>
675                 <h3 className='sub-headers'>Login</h3>
676                 <Form.Group controlId='username'>
677                     <Form.Label className='login-      ↗
labels'>User_ID</Form.Label>
678                     <Form.Control value={this.state.sql1}      ↗
type='username' placeholder="Enter SQL Injection here"      ↗
onChange={this.handleSQL1Change}></Form.Control>
679                 </Form.Group>
680                 <Button className='login-button'      ↗

```

```

        variant='primary' onClick={this.handleSQL1}>Login</Button>
681         <p> {this.state.sql1Response} </p>
682     </div>
683 </Form>
684 </Col>
685 </Row>
686 <p>SQL Batch Injection can be utilized to manipulate data
    outside of the confines of the table used in the login.
    Multiple SQL statements are joined together in order to
    retrieve sensitive information from the database.
687 In the example below, we're going to retrieve information
    related to "John Doe" that exists in the same schema as the
    previous example, but within a different table than the
    login Users table.
688 </p>
689 <br />
690 <p>The first SQL statement below is used to access the schema,
    while we retrieve the necessary server information with the
    second statement.Copy the combined statement into the
    following login box below:
691     <b>" ; SELECT Computer_Number FROM Computers WHERE
        Name = 'John Doe' --
692     </b>
693 </p>
694 <Row className="justify-content-md-center">
695     <Col xs={8}>
696         <Form>
697             <div align='center' className='login-form'>
698                 <h3 className='sub-headers'>Login</h3>
699                 <Form.Group controlId='username'>
700                     <Form.Label className='login-
labels'>User_ID</Form.Label>
701                     <Form.Control value={this.state.sql2}
type='username' placeholder="Enter SQL Batch injection
here" onChange={this.handleSQL2Change}></Form.Control>
702                 </Form.Group>
703                 <Button className='login-button'
variant='primary' onClick={this.handleSQL2}>Login</Button>
704                 <p> {this.state.sql2Response} </p>
705             </div>
706         </Form>
707     </Col>
708 </Row>
709 <br />
710 <br />
711 <p> <b>Good luck with the Mystery!</b></p>
712 <Button variant="outline-primary float-left" href="/rules"
>Back</Button>
713 <Button variant="outline-primary float-right" href="/"

```

```

    step1">Start Game!</Button>
714     </Container >
715
716     );
717   }
718 }
719
720 export default Practice;
721
722 //ResponseTable.js
723 //Andrew Fecher
724 export default class ResponseTable extends React.Component {
725   constructor(props) {
726     super(props);
727     this.state = {
728     }
729     this.props = props;
730   }
731
732   render() {
733     let sqltable = null;
734     if (this.props.results != '') {
735       var HTMLrows = [];
736       let obj = JSON.parse(this.props.results);
737       var header = [];
738       let ir = 0;
739       Object.keys(obj).forEach(function (rowInt) {
740         var rowData = obj[rowInt];
741         var HTMLrow = [];
742         Object.keys(rowData).forEach(function (colName) {
743           var value = rowData[colName];
744           if (ir == 0) {
745             header.push(<th>{colName}</th>)
746           }
747           HTMLrow.push(<td>{value}</td>);
748         });
749         HTMLrows.push(<tr>{HTMLrow}</tr>);
750         ir += 1;
751       });
752       sqltable = <table><tr>{header}</tr>{HTMLrows}</table>;
753     }
754     return (
755       <div align='center' className="sqltable">
756         {sqltable}
757       </div>
758     )
759   }
760 }
761

```

```

762 //rules.js
763 //Written by Julia Workum and Lia Ferguson
764
765 class Rules extends React.Component {
766     constructor(props) {
767         super(props);
768     }
769
770     render() {
771         return (
772             <Container fluid='md'>
773                 <h2 className='sub-headers'>Rules</h2>
774                 <p>
775                     This is an interactive game that will show you how SQL
776                     Injection can be used to exploit webpages in order to
777                     retrieve confidential information from their underlying
778                     database.
779                     You will also test your knowledge of SQL Injection and
780                     other important security topics through quiz-style
781                     questions and interactive activities.
782                     Each step of the game will require you to successfully
783                     execute SQL statements or prove your knowledge about SQL
784                     Injection to uncover clues that will help you to solve the
785                     unsolved murder of Tony Stark.
786                     There will be helpful hints along the way if you need help with the
787                     task at hand. <br></br> <br></br>
788
789                     As part of the game, text files may be downloaded onto your computer
790                     in order to relay important clues or information to you. They
791                     will be stored in a file called "SQL-Mystery-Game-Files" on your
792                     Desktop. <b> Good luck and happy hacking!</b>
793
794                 </p>
795                 <Button variant="outline-primary float-left" href="/" >Back</
796                 Button>
797                 <Button variant="outline-primary float-right" href="/practice" >Practice!</Button>
798             </Container>
799         );
800     }
801 }
802 export default Rules;
803
804 //SQLInput.js
805 //Julia Workum
806 class SQLInput extends React.Component {
807     constructor(props) {
808         super(props);

```

```
797     }
798
799     render() {
800         return (
801             // <Container fluid="md">
802             <Form>
803                 <Form.Group controlId="query">
804                     <Form.Label>Enter your SQL query below:</Form.Label>
805                     <Form.Control type="sql" placeholder="..." />
806                 </Form.Group>
807                 <Button variant="primary" type="submit">
808                     Run
809                 </Button>
810                 <Button variant="primary" type="submit">
811                     Reset
812                     { /* would want the input box cleared out on click */ }
813                 </Button>
814             </Form>
815             // </Container>
816         );
817     }
818 }
819
820 export default SQLInput;
821
822 //Step1.js
823 // Written by Lia Ferguson and Julia Workum
824
825 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
826
827 class Step1 extends React.Component {
828     constructor(props) {
829         super(props);
830
831         this.state = { isClicked: false, isQuerySuccessful: false, user_id:
832             '', password: '' };
833
834         this.handleQuery = this.handleQuery.bind(this);
835         this.handleUserIdChange = this.handleUserIdChange.bind(this);
836         this.handlePasswordChange = this.handlePasswordChange.bind(this);
837     }
838
839     handleUserIdChange(e) {
840         e.preventDefault()
841         this.setState({ user_id: e.target.value });
842     }
843
844     handlePasswordChange(e) {
845         e.preventDefault()
```

```
845     this.setState({ password: e.target.value });
846   }
847
848   async handleQuery() {
849     var response = await this.executeQuery(this.state.user_id,
850     this.state.password);
851     var isQuerySuccessful = response.isQuerySuccessful == 'true' ? true :
852     false
853     this.setState({ isQuerySuccessful: isQuerySuccessful });
854     this.setState({ isClicked: true })
855   }
856
857   async executeQuery(user_id, pwd, isQuerySuccessful) {
858     const response = await fetch(BACKEND_API_URL + "/login_bypass", {
859       method: "POST",
860       mode: 'cors',
861       headers: {
862         'Content-Type': 'application/json'
863       },
864       body: JSON.stringify({
865         isQuerySuccessful: isQuerySuccessful,
866         user_id: user_id,
867         password: pwd
868       })
869     })
870     return await response.json();
871   }
872
873   render() {
874     let queryResponse, continueButton;
875     if (this.state.isClicked && this.state.isQuerySuccessful) {
876       queryResponse = <div>
877         <p> Congratulations! You successfully bypassed authentication
878         by using SQL Injection!
879         If a website's backend does not sanitize user input before
880         using it in a SQL query,
881         you are able to "hijack" the query by placing a condition that
882         is always true into
883         the query in order to bypass the intended programatic flow.
884         </p>
885       </div>;
886       continueButton = <Button variant="outline-primary float-right"
887         href="/step2">Continue</Button>;
888     } else if (this.state.isClicked && !this.state.isQuerySuccessful) {
889       queryResponse = <div>
890         <p> Hmm, looks like your SQL Injection wasn't quite right.
891         Please try again.
892         Remember, use the hint if you are stumped!
893       </p>
894     }
```

```

887         </div>;
888         continueButton = null;
889     }
890     return (
891         <Container fluid='md'>
892             <h2 className='sub-headers'>Step One: Credential SQL Injection</h2>
893             <p>
894                 You now have Tony's computer, which you can use to view
895                 the company database. There's only one problem: you do not
896                 know his password to access the database. You can only
897                 think of one option to get access to his computer, SQL
898                 Injection! While it is a form of hacking, you deem it
899                 worthy in order to try to find out who murdered your
900                 friend, Tony.<br><br><br><br><b>Use SQL injection to
901                 bypass the authentication system.</b>
902             </p>
903             <Row className="justify-content-md-center">
904                 <Col xs={8}>
905                     <Accordion className='hint'>
906                         <Card>
907                             <Accordion.Toggle as={Card.Header}
908                                 eventKey="0">
909                                 Hint
910                             </Accordion.Toggle>
911                             <Accordion.Collapse eventKey="0">
912                                 <Card.Body>Think back to the practice
913                                 section you just completed. Rely on you knowledge of how
914                                 to write comments in SQL, and where the admin account is
915                                 usually stored in a database to write a query that will
916                                 bypass the login function. </Card.Body>
917                             </Accordion.Collapse>
918                         </Card>
919                     </Accordion>
920                 </Col>
921             </Row>
922             <Row className="justify-content-md-center">
923                 <Col xs={8}>
924                     <Form>
925                         <div align='center' className='login-form'>
926                             <h3 className='sub-headers'>Login</h3>
927                             <Form.Group controlId='username'>
928                                 <Form.Label className='login-
929                                     labels'>User_ID</Form.Label>
930                                 <Form.Control value={this.state.user_id}
931                                     type='username' placeholder="Enter User_ID here" onChange=
932                                     {this.handleUserIdChange}></Form.Control>
933                                 </Form.Group>
934                                 <Form.Group controlId='password'>

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 24
920         <Form.Label className='login-labels'  ↗
           >Password</Form.Label>
921         <Form.Control value={this.state.password}  ↗
           type='password' placeholder="Enter password here"  ↗
           onChange={this.handlePasswordChange} ></Form.Control>
922         </Form.Group>
923         <Button className='login-button'  ↗
           variant='primary' onClick={this.handleQuery}>Login</  ↗
           Button>
924         </div>
925         {queryResponse}
926         {/*<div className='sql-results'>
927         <ListGroup>
928         <ListGroup.Item> Results of SQL Injection</ListGroup.Item>
929         <ListGroup.Item>UserId = ...</ListGroup.Item>
930         <ListGroup.Item>Username = ...</ListGroup.Item>
931         <ListGroup.Item>Password = ...</ListGroup.Item>
932         </ListGroup>
933         </div>*/}
934         }
935         </Form>
936         </Col>
937         </Row>
938         <Row className="justify-content-md-center">
939         <Col xs={6}>
940         <Button variant="outline-primary float-left" href="/  ↗
           practice">Back</Button>
941         {continueButton}
942         </Col>
943         </Row>
944         </Container>
945
946     );
947 }
948 }
949
950 export default Step1;
951
952 //step2.js
953 //Julia Workum
954 class Step2 extends React.Component {
955     constructor(props) {
956         super(props);
957         this.state = { q1Correct: false, q2Correct: false, q3Correct: false,  ↗
           q4Correct: false, q5Correct: false, minutes: 2, seconds: 0 };
958         this.handleCorrectChoice = this.handleCorrectChoice.bind(this)
959     }
960
961     handleCorrectChoice(e) {

```



```
962     switch (e.target.value) {
963       case 'correct1':
964         this.setState({ q1Correct: true })
965         break;
966       case 'correct2':
967         this.setState({ q2Correct: true })
968         break;
969       case 'correct3':
970         this.setState({ q3Correct: true })
971         break;
972       case 'correct4':
973         this.setState({ q4Correct: true })
974         break;
975       case 'correct5':
976         this.setState({ q5Correct: true })
977         break;
978       default:
979         break;
980     };
981   }
982
983   componentDidMount() {
984     this.myInterval = setInterval(() => {
985       const { seconds, minutes } = this.state
986
987       if (seconds > 0) {
988         this.setState(({ seconds }) => ({
989           seconds: seconds - 1
990         }))
991       }
992       if (seconds === 0) {
993         if (minutes === 0) {
994           clearInterval(this.myInterval)
995         } else {
996           this.setState(({ minutes }) => ({
997             minutes: minutes - 1,
998             seconds: 59
999           }))
1000         }
1001       }
1002     }, 1000)
1003   }
1004
1005   componentWillUnmount() {
1006     clearInterval(this.myInterval)
1007   }
1008
1009   render() {
1010     const question1 = <p className="helper-text"><b>Question 1: </b>What ?
```

```

    is a honey pot?</p>;
1011     const questionAnswers1 = [
1012         <Card.Text className="answer-text"> The active screening of
            network and system activity for unauthorized access <br></br><br></br></Card.Text>,
1013         <Card.Text className="answer-text"> An attack on information
            assets in which attack agent gains or attempts to gain entry
            into a network or system in order to disrupt or cause other harm
            <br></br><br></br></Card.Text>,
1014         <Card.Text className="answer-text"> A series of steps or processes
            used by an attacker, in a logical sequence, to launch attack
            against a target system or network<br></br><br></br></Card.Text>,
1015         <Card.Text className="answer-text"> A decoy system designed to
            lure potential attackers away from critical systems and
            encourage attacks against themselves<br></br><br></br></Card.Text>
1016     ];
1017
1018     const questionHeaders1 = new Map();
1019     questionHeaders1['A'] = 'Incorrect';
1020     questionHeaders1['B'] = 'Incorrect';
1021     questionHeaders1['C'] = 'Incorrect';
1022     questionHeaders1['D'] = 'Correct';
1023
1024     const questionResponses1 = new Map();
1025     questionResponses1['A'] = 'No :( This is intrusion detection.';
1026     questionResponses1['B'] = 'Nope, that\'s an intrusion!';
1027     questionResponses1['C'] = 'No, that sounds like an attack protocol.';
1028     questionResponses1['D'] = 'Hmmm... that sounds kind of like this
            exercise...';
1029
1030     const question2 = <p className="helper-text" style={{ marginTop: 16 }}>
        <b>Question 2: </b>Which of the following best describes a padded
        cell?</p>;
1031     const questionAnswers2 = [
1032         <Card.Text className="answer-text"> A behavior-based IDPS that
            samples network activity and compares it with traffic that is
            known to be normal <br></br><br></br></Card.Text>,
1033         <Card.Text className="answer-text"> A type of honey pot that has
            increased protection so it cannot be compromised <br></br><br></br></Card.Text>,
1034         <Card.Text className="answer-text"> The systematic survey of all
            available applications (open ports) on all footprinted hosts
            <br></br><br></br></Card.Text>,
1035         <Card.Text className="answer-text"> A room with padded walls
            <br></br><br></br></Card.Text>
1036     ];
1037

```

```

1038 const questionHeaders2 = new Map();
1039 questionHeaders2['A'] = 'Incorrect';
1040 questionHeaders2['B'] = 'Correct';
1041 questionHeaders2['C'] = 'Incorrect';
1042 questionHeaders2['D'] = 'Incorrect';
1043
1044 const questionResponses2 = new Map();
1045 questionResponses2['A'] = 'This is actually a statistical anomaly - based IDPS.';
1046 questionResponses2['B'] = 'Great job!';
1047 questionResponses2['C'] = 'Try again! This is fingerprinting.';
1048 questionResponses2['D'] = 'Not quite.';
1049
1050 const question3 = <p className="helper-text" style={{ marginTop: 16 }} ><b>Question 3:</b> True or false, a honey net is a collection of honey pots connecting several honey pot systems on a subnet? </p>;
1051 const questionAnswers3 = [
1052   <Card.Text className="answer-text">True <br></br><br></br></Card.Text>,
1053   <Card.Text className="answer-text">False <br></br><br></br></Card.Text>
1054 ];
1055
1056 const questionHeaders3 = new Map();
1057 questionHeaders3['A'] = 'Correct';
1058 questionHeaders3['B'] = 'Incorrect';
1059
1060 const questionResponses3 = new Map();
1061 questionResponses3['A'] = 'Correct!';
1062 questionResponses3['B'] = 'Sorry, try again!';
1063
1064 const question4 = <p className="helper-text" style={{ marginTop: 16 }} ><b>Question 4:</b> While honey pots have many advantages, according to our textbook which of the following is not one? </p>;
1065 const questionAnswers4 = [
1066   <Card.Text className="answer-text">Attackers can be diverted to targets that they cannot damage.<br></br><br></br></Card.Text>,
1067   <Card.Text className="answer-text">Administrators have time to decide how to respond to an attacker.<br></br><br></br></Card.Text>,
1068   <Card.Text className="answer-text">The legal implications of using honeypots are well understood.<br></br><br></br></Card.Text>,
1069   <Card.Text className="answer-text">Attackers' actions can be easily and more extensively monitored with honeypots, and the records can be used to refine threat models and improve system protections.<br></br><br></br></Card.Text>
1070 ];
1071

```

```

1072     const questionHeaders4 = new Map();
1073     questionHeaders4['A'] = 'Incorrect';
1074     questionHeaders4['B'] = 'Incorrect';
1075     questionHeaders4['C'] = 'Correct';
1076     questionHeaders4['D'] = 'Incorrect';
1077
1078     const questionResponses4 = new Map();
1079     questionResponses4['A'] = 'Nope, this is actually one of the main      ↗
        benefits of using a honey pot.';
1080     questionResponses4['B'] = 'Not quite. This is a huge advantage of      ↗
        honeypots.';
1081     questionResponses4['C'] = 'Exactly. The legal implications of honey      ↗
        pots are complicated and unclear.';
1082     questionResponses4['D'] = 'Try again!';
1083
1084     const question5 = <p className="helper-text" style={{ marginTop: 16 }} ↗
        ><b>Question 5: </b> In what way were these last 5 questions a honey ↗
        pot?</p>;
1085     const questionAnswers5 = [
1086         <Card.Text className="answer-text"> We included a key              ↗
            characteristic of a honey pot, the countdown.<br></br><br></br></Card.Text>, ↗
1087         <Card.Text className="answer-text"> These questions distracted and ↗
            delayed you from achieving your actual goal.<br></br><br></br></Card.Text>, ↗
1088         <Card.Text className="answer-text"> This activity was not a honey ↗
            pot.<br></br><br></br></Card.Text>,
1089         <Card.Text className="answer-text"> We stole your identity while ↗
            you completed this exercise.<br></br><br></br></Card.Text>
1090     ];
1091
1092     const questionHeaders5 = new Map();
1093     questionHeaders5['A'] = 'Incorrect';
1094     questionHeaders5['B'] = 'Correct';
1095     questionHeaders5['C'] = 'Incorrect';
1096     questionHeaders5['D'] = 'Incorrect';
1097
1098     const questionResponses5 = new Map();
1099     questionResponses5['A'] = 'Nope. This is not part of a honey pot.';
1100     questionResponses5['B'] = 'Correct! You just wasted 2 minutes of your ↗
        life that you will never get back.';
1101     questionResponses5['C'] = 'No. Unfortunately it was a honey pot.';
1102     questionResponses5['D'] = 'Try again. Even if this was a              ↗
        characteristic of a honey pot, we this isn\'t something we did (as ↗
        far as you know...);
1103
1104     const { minutes, seconds } = this.state
1105
1106     let questionSetup1 = <Container>

```

```

1107     <Row className="justify-content-md-center" >
1108         {question1}
1109     </Row>
1110     <Row className="justify-content-md-center" >
1111         <CardGroup>
1112             <Card style={{ width: '18rem' }} bg="light">
1113                 <OverlayTrigger
1114                     trigger="click"
1115                     key='top'
1116                     rootClose
1117                     placement='top'
1118                     overlay={
1119                         <Popover id='popover-positioned-top'>
1120                             <Popover.Title as="h3">{questionHeaders1
1121                                 ['A']}</Popover.Title>
1122                             <Popover.Content>
1123                                 {questionResponses1['A']}
1124                             </Popover.Content>
1125                         </Popover>
1126                     }>
1127                     <Button className="question-button"
1128                         variant="outline-primary" size="sm"> A </Button>
1129                 </OverlayTrigger>
1130                 {questionAnswers1[0]}
1131             </Card>
1132             <Card style={{ width: '18rem' }} bg="light">
1133                 <OverlayTrigger
1134                     trigger="click"
1135                     key='top'
1136                     rootClose
1137                     placement='top'
1138                     overlay={
1139                         <Popover id='popover-positioned-top'>
1140                             <Popover.Title as="h3">{questionHeaders1
1141                                 ['B']}</Popover.Title>
1142                             <Popover.Content>
1143                                 {questionResponses1['B']}
1144                             </Popover.Content>
1145                         </Popover>
1146                     }>
1147                     <Button className="question-button"
1148                         variant="outline-primary" size="sm"> B </Button>
1149                 </OverlayTrigger>
1150                 {questionAnswers1[1]}
1151             </Card>
1152         </CardGroup>
1153     </Row>
1154     <Row className="justify-content-md-center" >
1155         <CardGroup>

```

```

1152         <Card style={{ width: '18rem' }} bg="light">
1153             <OverlayTrigger
1154                 trigger="click"
1155                 key='bottom'
1156                 rootClose
1157                 placement='bottom'
1158                 overlay={
1159                     <Popover id='popover-positioned-bottom'>
1160                         <Popover.Title as="h3">{questionHeaders1
1161                             ['C']}</Popover.Title>
1162                         <Popover.Content>
1163                             {questionResponses1['C']}
1164                         </Popover.Content>
1165                     </Popover>
1166                 }>
1167                 <Button className="question-button"
1168                     variant="outline-primary" size="sm"> C </Button>
1169             </OverlayTrigger>
1170             {questionAnswers1[2]}
1171         </Card>
1172     <Card style={{ width: '18rem' }} bg="light">
1173         <OverlayTrigger
1174             trigger="click"
1175             key='bottom'
1176             rootClose
1177             placement='bottom'
1178             overlay={
1179                 <Popover id='popover-positioned-top'>
1180                     <Popover.Title as="h3">{questionHeaders1
1181                         ['D']}</Popover.Title>
1182                     <Popover.Content>
1183                         {questionResponses1['D']}
1184                     </Popover.Content>
1185                 </Popover>
1186             }>
1187                 <Button className="question-button"
1188                     variant="outline-primary" size="sm" value='correct1'
1189                     onClick={this.handleCorrectChoice}> D </Button>
1190             </OverlayTrigger>
1191             {questionAnswers1[3]}
1192         </Card>
1193     </CardGroup>
1194 </Row>
1195 </Container>;
1196
1197 let startOverButton = null;
1198 let timer = <h1>Time remaining: {minutes}:{seconds < 10 ? `0${seconds}`
1199     : seconds}</h1>
1200 if (minutes === 0 && seconds === 0 && !this.state.q5Correct) {

```

```

1195         this.state.q1Correct = false;
1196         this.state.q2Correct = false;
1197         this.state.q3Correct = false;
1198         this.state.q4Correct = false;
1199         this.state.q5Correct = false;
1200         questionSetup1 = null;
1201         timer = null;
1202         startOverButton =
1203             <Container>
1204                 <Row className="justify-content-md-center"><h1>Time's up!
1205                 </h1></Row>
1206                 <Row className="justify-content-md-center"><Button
1207                     variant="danger float-right" href="/step1">Start Over</
1208                     Button></Row>
1209             </Container>
1210     } else if (minutes === 0 && seconds === 0 && this.state.q5Correct) {
1211         timer = <Row className="justify-content-md-center"><h1>Success!</
1212         h1></Row>
1213     }
1214
1215     let questionSetup2 = null;
1216     if (this.state.q1Correct) {
1217         questionSetup2 =
1218             <Container>
1219                 <Row className="justify-content-md-center">
1220                     {question2}
1221                 </Row>
1222                 <Row className="justify-content-md-center">
1223                     <CardGroup>
1224                         <Card style={{ width: '18rem' }} bg="light">
1225                             <OverlayTrigger
1226                                 trigger="click"
1227                                 key='top'
1228                                 rootClose
1229                                 placement='top'
1230                                 overlay={
1231                                     <Popover id='popover-positioned-top' >
1232                                         <Popover.Title as="h3">
1233                                             {questionHeaders2[ 'A' ]}</Popover.Title>
1234                                         <Popover.Content>
1235                                             {questionResponses2[ 'A' ]}
1236                                         </Popover.Content>
1237                                     </Popover>
1238                                 }>
1239                             <Button className="question-button"
1240                                 variant="outline-primary" size="sm"> A </Button>
1241                         </OverlayTrigger>
1242                         {questionAnswers2[0]}
1243                     </Card>

```

```

1238         <Card style={{ width: '18rem' }} bg="light">
1239             <OverlayTrigger
1240                 trigger="click"
1241                 key='top'
1242                 rootClose
1243                 placement='top'
1244                 overlay={
1245                     <Popover id='popover-positioned-top'>
1246                         <Popover.Title as="h3">
1247                             {questionHeaders2['B']}

```



```

1281         key= 'bottom'
1282         rootClose
1283         placement= 'bottom'
1284         overlay={
1285             <Popover id='popover-positioned-top' >
1286                 <Popover.Title as="h3">
1287                     {questionHeaders2[ 'D' ]}</Popover.Title>
1288                     <Popover.Content>
1289                         {questionResponses2[ 'D' ]}
1290                     </Popover.Content>
1291                 </Popover>
1292             </>
1293             <Button className="question-button"
1294                 variant="outline-primary" size="sm"> D </Button>
1295             </OverlayTrigger>
1296             {questionAnswers2[3]}
1297         </Card>
1298     </CardGroup>
1299 </Row>
1300 </Container>
1301 }
1302
1303 let questionSetup3 = null;
1304 if (this.state.q2Correct) {
1305     questionSetup3 =
1306     <Container>
1307         <Row className="justify-content-md-center">
1308             {question3}
1309         </Row>
1310         <Row className="justify-content-md-center">
1311             <CardGroup>
1312                 <Card style={{ width: '18rem' }} bg="light">
1313                     <OverlayTrigger
1314                         trigger="click"
1315                         key= 'top'
1316                         rootClose
1317                         placement= 'top'
1318                         overlay={
1319                             <Popover id='popover-positioned-top' >
1320                                 <Popover.Title as="h3">
1321                                     {questionHeaders3[ 'A' ]}</Popover.Title>
1322                                     <Popover.Content>
1323                                         {questionResponses3[ 'A' ]}
1324                                     </Popover.Content>
1325                                 </Popover>
1326                             </>
1327                             <Button className="question-button"
1328                                 variant="outline-primary" size="sm" value='correct3'
1329                                 onClick={this.handleCorrectChoice}> T </Button>

```



```

1371         </Popover.Content>
1372     </Popover>
1373 }>
1374     <Button className="question-button"
variant="outline-primary" size="sm"> A </Button>
1375     </OverlayTrigger>
1376     {questionAnswers4[0]}
1377 </Card>
1378 <Card style={{ width: '18rem' }} bg="light">
1379     <OverlayTrigger
1380         trigger="click"
1381         key='top'
1382         rootClose
1383         placement='top'
1384         overlay={
1385             <Popover id='popover-positioned-top' >
1386                 <Popover.Title as="h3">
{questionHeaders4['B']}</Popover.Title>
1387                 <Popover.Content>
1388                     {questionResponses4['B']}
1389                 </Popover.Content>
1390             </Popover>
1391         }>
1392         <Button className="question-button"
variant="outline-primary" size="sm"> B </Button>
1393         </OverlayTrigger>
1394         {questionAnswers4[1]}
1395     </Card>
1396 </CardGroup>
1397 </Row>
1398 <Row className="justify-content-md-center">
1399     <CardGroup>
1400         <Card style={{ width: '18rem' }} bg="light">
1401             <OverlayTrigger
1402                 trigger="click"
1403                 key='bottom'
1404                 rootClose
1405                 placement='bottom'
1406                 overlay={
1407                     <Popover id='popover-positioned-
bottom'>
1408                         <Popover.Title as="h3">
{questionHeaders4['C']}</Popover.Title>
1409                         <Popover.Content>
1410                             {questionResponses4['C']}
1411                         </Popover.Content>
1412                     </Popover>
1413                 }>
1414                 <Button className="question-button"

```

```

variant="outline-primary" size="sm" value='correct4'
onClick={this.handleCorrectChoice}> C </Button>
1415     </OverlayTrigger>
1416     {questionAnswers4[2]}
1417   </Card>
1418   <Card style={{ width: '18rem' }} bg="light">
1419     <OverlayTrigger
1420       trigger="click"
1421       key= 'bottom'
1422       rootClose
1423       placement= 'bottom'
1424       overlay={
1425         <Popover id='popover-positioned-top' >
1426           <Popover.Title as="h3">
{questionHeaders4[ 'D' ]}</Popover.Title>
1427           <Popover.Content>
1428             {questionResponses4[ 'D' ]}
1429           </Popover.Content>
1430         </Popover>
1431       }>
1432       <Button className="question-button"
variant="outline-primary" size="sm"> D </Button>
1433     </OverlayTrigger>
1434     {questionAnswers4[3]}
1435   </Card>
1436 </CardGroup>
1437 </Row>
1438 </Container>
1439 }
1440
1441 let questionSetup5 = null;
1442 if (this.state.q4Correct) {
1443   questionSetup5 =
1444     <Container>
1445       <Row className="justify-content-md-center">
1446         {question5}
1447       </Row>
1448       <Row className="justify-content-md-center">
1449         <CardGroup>
1450           <Card style={{ width: '18rem' }} bg="light">
1451             <OverlayTrigger
1452               trigger="click"
1453               key= 'top'
1454               rootClose
1455               placement= 'top'
1456               overlay={
1457                 <Popover id='popover-positioned-top' >
1458                   <Popover.Title as="h3">
{questionHeaders5[ 'A' ]}</Popover.Title>

```

```

1459         <Popover.Content>
1460             {questionResponses5[ 'A' ]}
1461         </Popover.Content>
1462     </Popover>
1463 }>
1464     <Button className="question-button"
variant="outline-primary" size="sm"> A </Button>
1465     </OverlayTrigger>
1466     {questionAnswers5[0]}
1467 </Card>
1468 <Card style={{ width: '18rem' }} bg="light">
1469     <OverlayTrigger
1470         trigger="click"
1471         key='top'
1472         rootClose
1473         placement='top'
1474         overlay={
1475             <Popover id='popover-positioned-top' >
1476                 <Popover.Title as="h3">
{questionHeaders5[ 'B' ]}</Popover.Title>
1477                 <Popover.Content>
1478                     {questionResponses5[ 'B' ]}
1479                 </Popover.Content>
1480             </Popover>
1481         }>
1482         <Button className="question-button"
variant="outline-primary" size="sm" value='correct5'
onClick={this.handleCorrectChoice}> B </Button>
1483     </OverlayTrigger>
1484     {questionAnswers5[1]}
1485 </Card>
1486 </CardGroup>
1487 </Row>
1488 <Row className="justify-content-md-center">
1489     <CardGroup>
1490         <Card style={{ width: '18rem' }} bg="light">
1491             <OverlayTrigger
1492                 trigger="click"
1493                 key='bottom'
1494                 rootClose
1495                 placement='bottom'
1496                 overlay={
1497                     <Popover id='popover-positioned-
bottom'>
1498                         <Popover.Title as="h3">
{questionHeaders5[ 'C' ]}</Popover.Title>
1499                         <Popover.Content>
1500                             {questionResponses5[ 'C' ]}
1501                         </Popover.Content>

```

```

1502         </Popover>
1503     }>
1504     <Button className="question-button"
variant="outline-primary" size="sm"> C </Button>
1505     </OverlayTrigger>
1506     {questionAnswers5[2]}
1507 </Card>
1508 <Card style={{ width: '18rem' }} bg="light">
1509     <OverlayTrigger
1510         trigger="click"
1511         key= 'bottom'
1512         rootClose
1513         placement= 'bottom'
1514         overlay={
1515             <Popover id='popover-positioned-top'>
1516                 <Popover.Title as="h3">
{questionHeaders5[ 'D' ]}</Popover.Title>
1517                 <Popover.Content>
1518                     {questionResponses5[ 'D' ]}
1519                 </Popover.Content>
1520             </Popover>
1521         }>
1522         <Button className="question-button"
variant="outline-primary" size="sm"> D </Button>
1523         </OverlayTrigger>
1524         {questionAnswers5[3]}
1525     </Card>
1526 </CardGroup>
1527 </Row>
1528 </Container>
1529 }
1530
1531 let continueButton = null;
1532 if (this.state.q5Correct) {
1533     continueButton = <Button variant="outline-primary float-right"
href="/step3">Continue</Button>
1534 }
1535
1536 return (
1537     <Container fluid='md'>
1538         <h2 className='sub-headers'>Step Two: You're in!</h2>
1539         <p>Congratulations, you're in! Now that you've bypassed Tony's
user authentication system it's time to answer a few
questions. However, in the spirit of Dr. Jones' quizzes,
you're being timed!<b> You'll have 2 minutes to answer 5
questions pertaining to what we've learned this semester in
CSE 4471.</b></p>
1540     <Row className="justify-content-md-center">
1541         <Col xs={8}>

```

```
1542         <div className="helper-text"> {timer} </div>
1543     </Col>
1544 </Row>
1545     {startOverButton}
1546     {questionSetup1}
1547     {questionSetup2}
1548     {questionSetup3}
1549     {questionSetup4}
1550     {questionSetup5}
1551     {continueButton}
1552     <Button variant="outline-primary float-left" href="/
        step1">Back</Button>
1553 </Container>
1554
1555     );
1556 }
1557 }
1558
1559 export default Step2;
1560
1561 //step3.js
1562 //Andrew Fecher
1563
1564 class Step3 extends React.Component {
1565     constructor(props) {
1566         super(props);
1567
1568         this.state = { isClicked: false, isSuccessful: false, batchSqlCorrect:
            false, step: 0, results: '', user_id: '', password: '' };
1569
1570         this.handleBatchQuerySuccess = this.handleBatchQuerySuccess.bind
            (this);
1571         this.processResults = this.processResults.bind(this);
1572         this.advance = this.advance.bind(this);
1573     }
1574
1575     handleBatchQuerySuccess(isSuccessful) {
1576         this.setState({ batchSqlCorrect: isSuccessful });
1577         if (this.state.step < 1) {
1578             this.setState({ step: 1 });
1579         }
1580     }
1581
1582     processResults(results) {
1583         this.setState({ results: results });
1584     }
1585
1586     advance() {
1587         if (this.state.step < 2) {
```

```

1588     this.setState({ step: 2 });
1589   }
1590 }
1591
1592 render() {
1593   let queryResponse = null, continueButton = null, table = null,
1594     nextButton = null;
1595   let loginSQL = <LoginSQL processResults={this.processResults}
1596     game_step='S3_B1' batchSqlCorrect={this.handleBatchQuerySuccess}
1597     congratsMessage="Congratulations, your SQL Injection was
1598     successful!" failureMessage="Hmm it doesn't look like your Injection
1599     Query was successful. Please try again."></LoginSQL>;
1600
1601   if (this.state.step == 1) {
1602     nextButton = <Button variant="outline-primary float-center"
1603       onClick={this.advance}>Got it!</Button>;
1604   }
1605   if (this.state.step > 0) {
1606     queryResponse = <div>
1607       <p> Logins do not directly output the results of thier queries
1608         to the front end, however they can be manipulated to do so
1609         using methods such as inducing errors, manipulating the
1610         outfile, sniffing, among many others. <br />
1611         Luckily many progams such as SQLMap and other will do this
1612         automatically. <br />
1613         The system being utilized here will do this automatically
1614         and just return the value of the second query made.
1615       </p>
1616       {nextButton}
1617     </div>;
1618   }
1619   if (this.state.step > 1) {
1620     table = <ResponseTable results={this.state.results} />;
1621     continueButton = continueButton = <Button variant="outline-primary
1622       float-right" href="/step4">Continue</Button>;
1623   }
1624
1625   return (
1626     <Container fluid='md'>
1627       <h2 className='sub-headers'>Step Three: Figure out the table
1628         names</h2>
1629       <p>In order to retrieve information from the database, you
1630         will need more information about the underlying database
1631         schema. <b>Use SQL injection to retrieve the names of the
1632         tables in the database.</b></p>
1633       <Row className="justify-content-md-center">
1634         <Col xs={8}>
1635           <Container>
1636             <h5>Table Names</h5>

```



```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 41
1621         <h6 className='sub-headers'> SQL Injection</h6>
1622         <p className="helper-text"><b>Use SQL Batch
Injection to find the names of all tables in the database
schema.</b></p>
1623         <Hint hint={ 'We know the database is running
SQLite, so you can use the "name" column of SQLite_master
to get the list of tables.' }></Hint>
1624         <Hint hint={ 'Make sure to only print out only the
type="table" entries.' }></Hint>
1625         {loginSQL}
1626         {queryResponse}
1627         {table}
1628     </Container>
1629 </Col>
1630 </Row>
1631 <Button variant="outline-primary float-left" href="/step2"
>Back</Button>
1632 {continueButton}
1633 </Container>
1634
1635     );
1636 }
1637 }
1638
1639 export default Step3;
1640
1641 //step4.js
1642 // Written by Lia Ferguson
1643
1644 class Step4 extends React.Component {
1645     constructor(props) {
1646         super(props);
1647         this.state = {
1648             clickCorrectTable: false, clickCorrectQInfo: false,
1649             batchSqlCorrect: false, correctQQ1: false, correctQQ2: false,
1650             batchSql2Correct: false,
1651             results1: '', results2: ''
1652         };
1653         this.handleClick = this.handleClick.bind(this);
1654         this.handleCorrectChoice = this.handleCorrectChoice.bind(this);
1655         this.handleBatchQuerySuccess = this.handleBatchQuerySuccess.bind
1656         (this);
1657         this.handleBatchQuery2Success = this.handleBatchQuery2Success.bind
1658         (this);
1659         this.processResults1 = this.processResults1.bind(this);
1660         this.processResults2 = this.processResults2.bind(this);
1661     }

```

```
1661     handleClick() {
1662         this.setState({ clickCorrectTable: true });
1663     }
1664
1665     handleCorrectChoice(e) {
1666         switch (e.target.value) {
1667             case 'infoQ':
1668                 this.setState({ clickCorrectQInfo: true });
1669                 break;
1670             case 'qq1':
1671                 this.setState({ correctQQ1: true });
1672                 break;
1673             case 'qq2':
1674                 this.setState({ correctQQ2: true });
1675                 break;
1676         };
1677     }
1678
1679     handleBatchQuerySuccess(isSuccessful) {
1680         this.setState({ batchSqlCorrect: isSuccessful });
1681     }
1682
1683     handleBatchQuery2Success(isSuccessful) {
1684         this.setState({ batchSql2Correct: isSuccessful });
1685     }
1686
1687     processResults1(results) {
1688         this.setState({ results1: results });
1689     }
1690
1691     processResults2(results) {
1692         this.setState({ results2: results });
1693     }
1694
1695     render() {
1696         const userQuestion = 'What is the best way to protect an application
1697                                from SQL Injection?';
1698         const userAnswers = [
1699             <Card.Text className="answer-text"> Encrypt all data stored in the
1700                database <br></br><br></br></Card.Text>,
1701             <Card.Text className="answer-text"> Limit admin access to as few
1702                users as possible <br></br><br></br></Card.Text>,
1703             <Card.Text className="answer-text"> Make a really complex schema
1704                that would be hard for an attacker to reverse engineer <br></
1705                br><br></br></Card.Text>,
1706             <Card.Text className="answer-text"> Sanitize all user input from
1707                form fields <br></br><br></br></Card.Text>
1708         ];
1709     }
1710 }
```

```

1704     let userHeader = new Map();
1705     userHeader['A'] = 'Incorrect';
1706     userHeader['B'] = 'Incorrect';
1707     userHeader['C'] = 'Incorrect';
1708     userHeader['D'] = 'Correct';
1709
1710     const userResponses = new Map();
1711     userResponses['A'] = 'Encrypting important data is good practice,
                                however encryption alone doesn\'t have a direct correlation to
                                preventing SQL Injection';
1712     userResponses['B'] = 'Limiting admin privileges is good practice, but
                                if admin credentials are stored in a database not protected against
                                SQL Injection, attackers can still exploit them.';
1713     userResponses['C'] = 'Having a complex schema might limit the amount
                                of info that could be retrieved using SQL Injection, but it isn\'t
                                the best countermeasure';
1714     userResponses['D'] = 'Sanitizing all user input ensures that any
                                unexpected input either won\'t be accepted by your form, or it won
                                \'t be executed as a SQL query, protecting your database.';
1715
1716     const qq1Question = 'What is one way to test if an application is
                                vulnerable to SQL Injection?';
1717
1718     const qq1Answers = [
1719         <Card.Text className="answer-text"> Enter random expected values
                                into the form field <br></br><br></br></Card.Text>,
1720         <Card.Text className="answer-text"> Enter a single or double quote
                                into the form fields to see if there is an internal server
                                error or other unexpected error message <br></br><br></br></
                                Card.Text>,
1721         <Card.Text className="answer-text"> Open a website in multiple
                                browsers to look for visible differences
                                <br></br><br></br></Card.Text>,
1722         <Card.Text className="answer-text"> Use a packet sniffer to
                                examine packets coming in and out of network ports related to
                                the application
                                <br></br><br></br></Card.Text>
1724     ];
1725
1726     const qq1Header = new Map();
1727     qq1Header['A'] = 'Incorrect';
1728     qq1Header['B'] = 'Correct';
1729     qq1Header['C'] = 'Incorrect';
1730     qq1Header['D'] = 'Incorrect';
1731
1732     const qq1Responses = new Map();
1733     qq1Responses['A'] = 'Even though values are being guessed, this is
                                really just using the website as it was intended';
1734     qq1Responses['B'] = 'Internal server errors or generic error messages

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 44
    after a single or double quote often mean that user input isn't
    being sanitized properly and SQL Injection might be possible';
1736 qq1Responses['C'] = 'Vulnerability to SQL Injection can't be
    identified just by observing the frontend design of an application';
1737 qq1Responses['D'] = 'This technique could be used to find out
    information about the database that is used, but it won't
    necessarily tell you if the application itself is vulnerable to SQL
    Injection';
1738
1739 const qq2Question = 'What data properties are most at risk from SQL
    Injection?';
1740
1741 const qq2Answers = [
1742     <Card.Text className="answer-text"> Confidentiality and Integrity
        <br></br><br></br></Card.Text>,
1743     <Card.Text className="answer-text"> Authenticity and Utility
        <br></br><br></br></Card.Text>
1744 ];
1745
1746 const qq2Header = new Map();
1747 qq2Header['A'] = 'Correct';
1748 qq2Header['B'] = 'Incorrect';
1749
1750 const qq2Responses = new Map();
1751 qq2Responses['A'] = 'SQL Injection exposes confidential data from an
    application's databases to threat actors, and it is possible for
    attackers to delete or change data using injection, thus
    compromising data's confidentiality and integrity.';
1752 qq2Responses['B'] = 'SQL Injection doesn't necessarily change the
    authenticity of data, it makes authentic data accessible by
    attackers. The utility of data is extremely high in the hands of an
    attacker and remains high for the application it belongs to.';
1753
1754
1755 let userIdColumnText = null;
1756 let userIdQuestion = null;
1757 if (this.state.clickCorrectTable) {
1758     userIdColumnText = <div>
1759         <p>
1760             As a backup, Stark Industries keeps a collection of quick
            access information that database
1761             administrators can use in order to refresh their memory
            about the database without having
1762             to sift through the large schema. To safeguard this
            information, database administrators are required to
1763             prove their security knowledge to ensure that only
            trusted, skilled professionals have access to the info.
1764             <br></br><br></br>
1765             </p>

```



```

1810         </Popover>
1811     }>
1812     <Button className="question-button"
variant="outline-primary" size="sm"> B </Button>
1813     </OverlayTrigger>
1814     {userAnswers[1]}
1815 </Card>
1816 </CardGroup>
1817 </Row>
1818 <Row className="justify-content-md-center">
1819     <CardGroup>
1820         <Card style={{ width: '18rem' }} bg="light">
1821             <OverlayTrigger
1822                 trigger="click"
1823                 key= 'bottom'
1824                 rootClose
1825                 placement= 'bottom'
1826                 overlay={
1827                     <Popover id='popover-positioned-bottom' >
1828                         <Popover.Title as="h3">{userHeader
[ 'C' ]}</Popover.Title>
1829                         <Popover.Content>
1830                             {userResponses[ 'C' ]}
1831                         </Popover.Content>
1832                     </Popover>
1833                 }>
1834                 <Button className="question-button"
variant="outline-primary" size="sm"> C </Button>
1835                 </OverlayTrigger>
1836                 {userAnswers[2]}
1837             </Card>
1838             <Card style={{ width: '18rem' }} bg="light">
1839                 <OverlayTrigger
1840                     trigger="click"
1841                     key= 'bottom'
1842                     rootClose
1843                     placement= 'bottom'
1844                     overlay={
1845                         <Popover id='popover-positioned-top' >
1846                             <Popover.Title as="h3">{userHeader
[ 'D' ]}</Popover.Title>
1847                             <Popover.Content>
1848                                 {userResponses[ 'D' ]}
1849                             </Popover.Content>
1850                         </Popover>
1851                     }>
1852                     <Button className="question-button"
variant="outline-primary" size="sm" value='infoQ' onClick=
{this.handleCorrectChoice}> D </Button>

```

```

1853         </OverlayTrigger>
1854         {userAnswers[3]}
1855     </Card>
1856 </CardGroup>
1857 </Row>
1858 </Container>;
1859
1860 }
1861 let batchInjectInstructions = null;
1862 let batchInjectSection = null;
1863 let batchInjectFinal = null;
1864 let table1 = null;
1865 if (this.state.clickCorrectQInfo) {
1866     batchInjectInstructions = <div className="instruction-div">
1867         <p className="helper-text"><b>CLUE:</b> The names of the
1868             columns you will need are: User_ID, First_name, Last_name </p>
1869         <h6 className="sub-headers"> SQL Injection</h6>
1870         <p className="helper-text"><b>Use Batch SQL Injection to
1871             retrieve all name and User ID records from the USER_INFO
1872             table. Make special note of Tony's ID. </b></p>
1873     </div>;
1874     batchInjectSection = <div>
1875         <Hint hint={"Batch injection is performed by completing the
1876             expected query and ending it with a semi colon, and then
1877             typing another query following it that will retrieve the
1878             information you desire from the database."}></Hint>
1879         <LoginSQL game_step='S4_B1' processResults=
1880             {this.processResults1} batchSqlCorrect=
1881             {this.handleBatchQuerySuccess}
1882             congratsMessage="Congratulations, your SQL Injection was
1883             successful! Here are the results of your query:"
1884             failureMessage="Hmm it doesn't look like your Injection
1885             Query was successful. Please try again."></LoginSQL>
1886     </div>
1887     table1 = <ResponseTable results={this.state.results1} />;
1888 }
1889 let questionnaireBackground = null;
1890 let questionnaireQuestions = null;
1891
1892 if (this.state.batchSqlCorrect) {
1893     batchInjectFinal = <div className='text-under-table'>
1894         <p className="helper-text">Well done! Now we know how to
1895             identify Tony's information in all of the database tables. </p>
1896     </div>;
1897     questionnaireBackground = <div>
1898         <h5>Background</h5>
1899         <p>

```

```

1887 The police report also mentioned that Pepper Pots sent out
1888 a questionnaire to all the company's employees
1889 to collect personal information from employees in order to
1890 tailor events to their food and activity preferences.
1891 Pepper kept a copy of the questionnaire responses in a
1892 text document in her company account in addition to the
1893 database copy,
1894 and police suspect that the murderer hacked into her
1895 account to retrieve the personal information about Tony so
1896 they
1897 could plot their crime. <br><br><br><br>
1898 </p>
1899 <p className="helper-text">
1900 <b>Answer the following security questions to reveal the
1901 column names you need to query the QUESTIONNAIRE table for
1902 Tony's responses.</b>
1903 </p>
1904 </div>;
1905 questionnaireQuestions = <Container>
1906 <Container>
1907 <Row>
1908 <p>
1909 <b>Question</b><br><br>
1910 {qq1Question}
1911 </p>
1912 </Row>
1913 <Row >
1914 <Col>
1915 <CardGroup>
1916 <Card style={{ width: '18rem' }} bg="light">
1917 <OverlayTrigger
1918 trigger="click"
1919 key='top'
1920 rootClose
1921 placement='top'
1922 overlay={
1923 <Popover id='popover-positioned-
1924 top'>
1925 <Popover.Title as="h3">
1926 {qq1Header[ 'A' ]}</Popover.Title>
1927 <Popover.Content>
1928 {qq1Responses[ 'A' ]}
1929 </Popover.Content>
1930 </Popover>
1931 }>
1932 <Button className="question-button"
1933 variant="outline-primary" size="sm"> A </Button>
1934 </OverlayTrigger>
1935 {qq1Answers[0]}

```



```

1925         </Card>
1926         <Card style={{ width: '18rem' }} bg="light">
1927             <OverlayTrigger
1928                 trigger="click"
1929                 key='top'
1930                 rootClose
1931                 placement='top'
1932                 overlay={
1933                     <Popover id='popover-positioned-
top'>
1934                         <Popover.Title as="h3">
{qq1Header[ 'B' ]}</Popover.Title>
1935                         <Popover.Content>
1936                             {qq1Responses[ 'B' ]}
1937                         </Popover.Content>
1938                     </Popover>
1939                 }>
1940                 <Button className="question-button"
variant="outline-primary" size="sm" value="qq1" onClick=
{this.handleCorrectChoice}> B </Button>
1941             </OverlayTrigger>
1942             {qq1Answers[1]}
1943         </Card>
1944     </CardGroup>
1945 </Col>
1946 <Col>
1947     <div className="col-table">
1948         <Row className="justify-content-md-center">
1949             <h6 className="sub-headers">Column Names</
h6>
1950         </Row>
1951         <Row className="justify-content-md-center">
1952             <table>
1953                 <tr>
1954                     <th>Column 1</th>
1955                     <th>Column 2</th>
1956                     <th>Column 3</th>
1957                     <th>Column 4</th>
1958                 </tr>
1959                 <tr>
1960                     <td>{this.state.correctQQ1 ?
'Favorite_food' : '?'}</td>
1961                     <td>{this.state.correctQQ1 ?
'Favorite_hobby' : '?'}</td>
1962                     <td>{this.state.correctQQ2 ?
'Favorite_drink' : '?'}</td>
1963                     <td>{this.state.correctQQ2 ?
'Allergies' : '?'}</td>
1964                 </tr>

```

```

1965         </table>
1966     </Row>
1967 </div>
1968 </Col>
1969 </Row>
1970 <Row>
1971     <CardGroup className="second-row-qq">
1972         <Card style={{ width: '16rem' }} bg="light">
1973             <OverlayTrigger
1974                 trigger="click"
1975                 key='bottom'
1976                 rootClose
1977                 placement='bottom'
1978                 overlay={
1979                     <Popover id='popover-positioned-
bottom'>
1980                         <Popover.Title as="h3">{qq1Header
['C']}</Popover.Title>
1981                         <Popover.Content>
1982                             {qq1Responses['C']}
1983                         </Popover.Content>
1984                     </Popover>
1985                 }>
1986                 <Button className="question-button"
variant="outline-primary" size="sm"> C </Button>
1987             </OverlayTrigger>
1988             {qq1Answers[2]}
1989         </Card>
1990         <Card style={{ width: '16rem' }} bg="light">
1991             <OverlayTrigger
1992                 trigger="click"
1993                 key='bottom'
1994                 rootClose
1995                 placement='bottom'
1996                 overlay={
1997                     <Popover id='popover-positioned-
bottom'>
1998                         <Popover.Title as="h3">{qq1Header
['D']}</Popover.Title>
1999                         <Popover.Content>
2000                             {qq1Responses['D']}
2001                         </Popover.Content>
2002                     </Popover>
2003                 }>
2004                 <Button className="question-button"
variant="outline-primary" size="sm"> D </Button>
2005             </OverlayTrigger>
2006             {qq1Answers[3]}
2007         </Card>

```

```

2008      </CardGroup>
2009    </Row>
2010    <br></br> <br></br>
2011    <Row>
2012      <div className="second-row-qq">
2013        <Col>
2014          <Row>
2015            <p>
2016              <b>Question</b><br></br>
2017              {qq2Question}
2018            </p>
2019          </Row>
2020          <Row>
2021            <CardGroup>
2022              <Card style={{ width: '16rem' }}
2023                bg="light">
2024                <OverlayTrigger
2025                  trigger="click"
2026                  key='top'
2027                  rootClose
2028                  placement='top'
2029                  overlay={
2030                    <Popover id='popover-
2031                      positioned-top'>
2032                      <Popover.Title
2033                        as="h3">{qq2Header[ 'A' ]}</Popover.Title>
2034                      <Popover.Content>
2035                        {qq2Responses
2036                          [ 'A' ]}
2037                      </Popover.Content>
2038                    </Popover>
2039                  }>
2040                    <Button className="question-
2041                      button" variant="outline-primary" size="sm" value="qq2"
2042                      onClick={this.handleCorrectChoice} > A </Button>
2043                  </OverlayTrigger>
2044                  {qq2Answers[0]}
2045                </Card>
2046                <Card style={{ width: '16rem' }}
2047                  bg="light">
2048                  <OverlayTrigger
2049                    trigger="click"
2050                    key='top'
2051                    rootClose
2052                    placement='top'
2053                    overlay={
2054                      <Popover id='popover-
2055                        positioned-top'>
2056                        <Popover.Title

```

```

as="h3">{qq2Header[ 'B' ]}</Popover.Title>
2049                                     < Popover.Content>
2050                                     {qq2Responses
                                     ['B' ]}
2051                                     </ Popover.Content>
2052                                 </ Popover>
2053                             }>
2054                             <Button className="question-
button" variant="outline-primary" size="sm"> B </Button>
2055                             </ OverlayTrigger>
2056                             {qq2Answers[1]}
2057                         </ Card>
2058                     </ CardGroup>
2059                 </ Row>
2060             </ Col>
2061         </ div>
2062     </ Row>
2063 </ Container>
2064 </ Container>;
2065 }
2066 let batchInjectInstructions2 = null;
2067 let batchInjectSection2 = null;
2068 let table2 = null;
2069 if (this.state.correctQQ2) {
2070     batchInjectInstructions2 = <div className="instruction-div">
2071         <p className="helper-text">Great! Now you have all of the
information you need to find Tony's questionnaire
information. </p>
2072         <h6 className="sub-headers"> SQL Injection</h6>
2073         <p className="helper-text"><b>Use Batch SQL Injection with the
column names above to retrieve Tony Stark's Questionnaire
information from the QUESTIONNAIRE table.</b></p>
2074     </div>;
2075     batchInjectSection2 = <div>
2076         <Hint hint={"Use the same SQL techniques you used for the
first Batch Injection problem, just substitute in the
QUESTIONNAIRE table information."}></Hint>
2077         <LoginSQL game_step='S4_B2' processResults=
{this.processResults2} batchSqlCorrect=
{this.handleBatchQuery2Success}
congratsMessage="Congratulations, your SQL Injection was
successful! Here are the results of your query:"
failureMessage="Hmm it doesn't look like your Injection
Query was successful. Please try again."></LoginSQL>
2078     </div>;
2079     table2 = <ResponseTable results={this.state.results2} />;
2080 }
2081
2082 let batchFileOutput = null;

```

```

2083     let continueButton = null;
2084     if (this.state.batchSql2Correct) {
2085         batchFileOutput = <div>
2086             <p className="text-under-table helper-text"> The output of
                this query was also saved in a text file called "Clues.txt"
                in the SQL-Mystery-Game-Files folder on your Desktop for
                future reference.</p>
2087         </div>;
2088         continueButton = <Button variant="outline-primary float-right"
                href="/step5">Continue</Button>;
2089     }
2090
2091
2092     return (
2093         <Container fluid="md">
2094             <h2 className='sub-headers'>Step Four: Security Knowledge and
                Targeted SQL</h2>
2095             <h5>Background</h5>
2096             <p>
2097                 From the police report, we know that Tony died from a
                reaction to some unknown substance. In order to get
2098                 more insight into Tony's life for clues that might help
                us, let's query some of the database tables referenced
                in the partial schema we retrieved.
2099             </p>
2100             <h5>Employee User ID</h5>
2101             <p>
2102                 Everyone in the company has a unique user ID number that
2103                 is used to link all of their data back to them.
2104                 Based on the table names, which table should we use to
                find Tony's ID?
2105             </p>
2106             <h6 className="sub-headers">Table Names</h6>
2107             <br />
2108             <ButtonGroup className='mb-2'>
2109                 <OverlayTrigger
2110                     trigger="click"
2111                     key='top'
2112                     rootClose
2113                     placement='top'
2114                     overlay={
2115                         <Popover id='popover-positioned-top'>
2116                             <Popover.Title as="h3">Incorrect</
                Popover.Title>
2117                             <Popover.Content>Trial and error when
                navigating a schema blind is completely normal! Please try
                again! </Popover.Content>
2118                         </Popover>
2119                     }>

```

```
2120         <Button variant='outline-primary'>USERS</Button>
2121     </OverlayTrigger>
2122     <OverlayTrigger
2123         trigger="click"
2124         key='top'
2125         rootClose
2126         placement='top'
2127         overlay={
2128             <Popover id='popover-positioned-top'>
2129                 <Popover.Title as="h3">Correct!</
2130                 Popover.Title>
2131             </Popover>
2132             <Button variant='outline-primary' onClick=
2133             {this.handleClick}>USER_INFO</Button>
2134         </OverlayTrigger>
2135         <OverlayTrigger
2136             trigger="click"
2137             key='top'
2138             rootClose
2139             placement='top'
2140             overlay={
2141                 <Popover id='popover-positioned-top'>
2142                     <Popover.Title as="h3">Incorrect</
2143                     Popover.Title>
2144                     <Popover.Content>Trial and error when
2145                     navigating a schema blind is completely normal! Please try
2146                     again!</Popover.Content>
2147                 </Popover>
2148             }>
2149             <Button variant='outline-primary'>QUESTIONNAIRE</
2150             Button>
2151         </OverlayTrigger>
2152         <OverlayTrigger
2153             trigger="click"
2154             key='top'
2155             rootClose
2156             placement='top'
2157             overlay={
2158                 <Popover id='popover-positioned-top'>
2159                     <Popover.Title as="h3">Incorrect</
2160                     Popover.Title>
2161                     <Popover.Content>Trial and error when
2162                     navigating a schema blind is completely normal! Please try
2163                     again!</Popover.Content>
2164                 </Popover>
2165             }>
2166             <Button variant='outline-primary'>PURCHASE_ORDERS</
2167             Button>
```

```

2159         </OverlayTrigger>
2160         <OverlayTrigger
2161             trigger="click"
2162             key='top'
2163             rootClose
2164             placement='top'
2165             overlay={
2166                 <Popover id='popover-positioned-top'>
2167                     <Popover.Title as="h3">Incorrect</
2168                         Popover.Title>
2169                     <Popover.Content>Trial and error when
2170                         navigating a schema blind is completely normal! Please try
2171                         again!</Popover.Content>
2172                     </Popover>
2173                 }>
2174                 <Button variant='outline-primary'>BUILDING_ACCESS</
2175                     Button>
2176             </OverlayTrigger>
2177         </ButtonGroup>
2178         <br />
2179         <br />
2180         <br />
2181         {userIdColumnText}
2182         {userIdQuestion}
2183         {batchInjectInstructions}
2184         {batchInjectSection}
2185         {table1}
2186         {batchInjectFinal}
2187         {questionnaireBackground}
2188         {questionnaireQuestions}
2189         {batchInjectInstructions2}
2190         {batchInjectSection2}
2191         {table2}
2192         {batchFileOutput}
2193         <Button variant="outline-primary float-left" href="/step3"
2194             >Back</Button>
2195         {continueButton}
2196     </Container>
2197 );
2198 }
2199 }
2200 export default Step4;
2201 //step6.js
2202 //Written by Lia Ferguson
2203 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
2204
2205 class Step6 extends React.Component {
2206     constructor(props) {

```

```
2203     super(props);
2204
2205     this.state = {
2206         batch1Correct: false, batch2Correct: false, batch3Correct: false, ↗
2207         confirmSuspectCorrect: false,
2208         suspect1Correct: false, suspect2Correct: false, results1: '', ↗
2209         results2: '', results3: ''
2210     };
2211
2212     this.handleBatch1Success = this.handleBatch1Success.bind(this);
2213     this.handleBatch2Success = this.handleBatch2Success.bind(this);
2214     this.handleBatch3Success = this.handleBatch3Success.bind(this);
2215     this.handleConfirmSuspect = this.handleConfirmSuspect.bind(this);
2216     this.handleSuspect1Correct = this.handleSuspect1Correct.bind(this);
2217     this.handleSuspect2Correct = this.handleSuspect2Correct.bind(this);
2218     this.processResults1 = this.processResults1.bind(this);
2219     this.processResults2 = this.processResults2.bind(this);
2220     this.processResults3 = this.processResults3.bind(this);
2221
2222     handleBatch1Success() {
2223         this.setState({ batch1Correct: true });
2224     }
2225
2226     handleBatch2Success() {
2227         this.setState({ batch2Correct: true });
2228     }
2229
2230     handleBatch3Success() {
2231         this.setState({ batch3Correct: true });
2232     }
2233
2234     handleConfirmSuspect(isCorrect) {
2235         this.setState({ confirmSuspectCorrect: isCorrect });
2236     }
2237
2238     handleSuspect1Correct(isCorrect) {
2239         this.setState({ suspect1Correct: isCorrect });
2240     }
2241
2242     handleSuspect2Correct(isCorrect) {
2243         this.setState({ suspect2Correct: isCorrect });
2244     }
2245
2246     processResults1(results) {
2247         this.setState({ results1: results });
2248     }
2249
2250     processResults2(results) {
```



```

2250     this.setState({ results2: results });
2251   }
2252
2253   processResults3(results) {
2254     this.setState({ results3: results });
2255   }
2256   render() {
2257     let table1, injection2, confirmSuspect = null;
2258     if (this.state.batch1Correct) {
2259       table1 = <ResponseTable results={this.state.results1} />;
2260       injection2 = <Container>
2261         <h5>Check Suspect Building Access</h5>
2262         <p>Great! Now you have enough information to check when
2263           Natasha and Bruce entered the building. Remember, you have
2264           their User_ID's in your 'Clues.txt' file</p>
2265         <h6 className='sub-headers'> SQL Injection 2</h6>
2266         <p className='helper-text'><b>Use SQL Injection to find out
2267           what time Natasha and Bruce accessed the building on the day
2268           of Tony's death. Remember to return the User_ID column as
2269           well so you know who entered the building when!</b></p>
2270         <LoginSQL game_step='S6_B2' processResults=
2271           {this.processResults2} batchSqlCorrect=
2272           {this.handleBatch2Success} congratsMessage="Congratulations,
2273           your SQL Injection was successful! Here are the results of
2274           your query:" failureMessage="Hmm it doesn't look like your
2275           Injection Query was successful. Please try again."></
2276           LoginSQL>
2277         </Container>;
2278       }
2279
2280     let table2 = null;
2281     if (this.state.batch2Correct) {
2282       table2 = <ResponseTable results={this.state.results2} />;
2283       confirmSuspect = <ConfirmSuspect suspectConfirmCorrect=
2284         {this.handleConfirmSuspect}></ConfirmSuspect>;
2285     }
2286
2287     let injection3, table3 = null;
2288     if (this.state.confirmSuspectCorrect) {
2289       injection3 = <Container>
2290         <h5>New Suspect Building Access</h5>
2291         <p className='helper-text'>Hmm... it seems that both Natasha
2292           or Bruce didn't check into the building until <b><i>after</
2293           i></b> Tony was found in the break room.
2294           Looks like the lead on liking almonds wasn't quite as
2295           fruitful as we hoped. Let's pivot and find out who was in
2296           the building before Tony to gain our next round of
2297           suspects.</p>
2298         <h6 className='sub-headers'> SQL Injection 3</h6>

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 58
2282     <p className='helper-text'><b>Use SQL Injection to find out  ↗
        who accessed the building before Tony. Remember, that would  ↗
        be anyone who entered the building before 11:30 am! </b></p>
2283     <LoginSQL game_step='S6_B3' processResults=  ↗
        {this.processResults3} batchSqlCorrect=  ↗
        {this.handleBatch3Success} congratsMessage="Congratulations,  ↗
        your SQL Injection was successful! Here are the results of  ↗
        your query:" failureMessage="Hmm it doesn't look like your  ↗
        Injection Query was successful. Please try again."></  ↗
        LoginSQL>
2284     </Container>
2285     table3 = <ResponseTable results={this.state.results3} />;
2286 }
2287
2288 let suspects = null;
2289 if (this.state.batch3Correct) {
2290     suspects = <Container>
2291         <p className="text-under-table helper-text">The outputs from  ↗
            the past three queries are saved in the Clues.txt file for  ↗
            future reference.</p>
2292         <h5>New Suspects</h5>
2293         <p> A new lead! It looks like there are two employees who  ↗
            entered the building before Tony on the day he died. </p>
2294         <h6 className='sub-headers'> Declare Suspects</h6>
2295         <p className="helper-text"><b>Check the list of user data in  ↗
            your list of clues, find the names of employees who entered  ↗
            the building before 11:30am by comparing the User_IDs, and  ↗
            enter them in the fields below.</b></p>
2296         <Suspect game_step={'S6_S'} suspectCorrect=  ↗
            {this.handleSuspect1Correct}></Suspect>
2297         <Suspect game_step={'S6_S'} suspectCorrect=  ↗
            {this.handleSuspect2Correct}></Suspect>
2298         <p className="helper-text">These suspect names will be saved  ↗
            in the Clues.txt file for future reference.</p>
2299     </Container>;
2300 }
2301
2302 let continueButton = (this.state.suspect1Correct &&  ↗
    this.state.suspect2Correct) ? <Button variant="outline-primary  ↗
    float-right" href="/step7">Continue</Button> : null;
2303
2304 return (
2305     <Container fluid="md">
2306         <h2 className='sub-headers'>Step Six: Targeted SQL</h2>
2307         <h5>Background</h5>
2308         <p>
2309             It looks like Natasha Romanoff and Bruce Banner are your  ↗
            prime suspects so far. It seems hard to believe that they  ↗
            would turn on Tony Because

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 59
2310         they are all such good friends but... Natasha is a trained  ↗
        assassin and the big green guy can be pretty  ↗
        unpredictable. Let's see if you can find any
2311         additional evidence that will show you whether you're on  ↗
        the right trail or not. <br></br><br></br>The police  ↗
        report states that Tony was last seen entering the  ↗
        building at 11:30am before he was found in the break room  ↗
        at 12:30pm.
2312         It's likely that the murderer entered the building before  ↗
        Tony in order to stage their crime.
2313     </p>
2314     <h5>BUILDING_ACCESS Table Information</h5>
2315     <p>
2316         First, you need more information about the BUILDING_ACCESS  ↗
        table in order to query for specific information about  ↗
        your suspects. In SQLite, the underlying database system  ↗
        of the company,
2317         the function PRAGMA_TABLE_INFO(table_name) returns  ↗
        information about the given table in the database schema.  ↗
        The 'name' property will return the names of the table  ↗
        columns.
2318     </p>
2319     <h6 className='sub-headers'>SQL Injection 1</h6>
2320     <p className='helper-text'><b>Use Batch SQL Injection and the  ↗
        PRAGMA_TABLE_INFO function to determine the column names of  ↗
        the BUILDING_ACCESS table.</b></p>
2321     <Hint hint={"Use any valid statement to finish the expected  ↗
        query. Use a SELECT statement with the PRAGMA function to  ↗
        find the column names. "}></Hint>
2322     <LoginSQL game_step='S6_B1' processResults=  ↗
        {this.processResults1} batchSqlCorrect=  ↗
        {this.handleBatch1Success} congratsMessage="Congratulations,  ↗
        your SQL Injection was successful! Here are the results of  ↗
        your query:" failureMessage="Hmm it doesn't look like your  ↗
        Injection Query was successful. Please try again."></  ↗
        LoginSQL>
2323     {table1}
2324     {injection2}
2325     {table2}
2326     {confirmSuspect}
2327     {injection3}
2328     {table3}
2329     {suspects}
2330     <Button variant="outline-primary float-left" href="/step5"  ↗
        >Back</Button>
2331     {continueButton}
2332 </Container>
2333 );
2334 }

```

```
2335 }
2336
2337 export default Step6;
2338
2339 //step7.js
2340 // Written by Lia Ferguson
2341
2342 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
2343
2344 class Step7 extends React.Component {
2345   constructor(props) {
2346     super(props);
2347
2348     this.state = {
2349       batch1Correct: false, batch2Correct: false, user_id: '', password: ''
2350       isLoginSuccessful: false, errorMessage: '', loginClicked: false,
2351       suspectCorrect: false,
2352       results1: '', results2: ''
2353     };
2354
2355     this.handleBatch1Success = this.handleBatch1Success.bind(this);
2356     this.handleBatch2Success = this.handleBatch2Success.bind(this);
2357     this.handleQuery = this.handleQuery.bind(this);
2358     this.handleSuspectCorrect = this.handleSuspectCorrect.bind(this);
2359     this.handleUserIdChange = this.handleUserIdChange.bind(this);
2360     this.handlePasswordChange = this.handlePasswordChange.bind(this);
2361     this.processResults1 = this.processResults1.bind(this);
2362     this.processResults2 = this.processResults2.bind(this);
2363   }
2364
2365   handleBatch1Success() {
2366     this.setState({ batch1Correct: true });
2367   }
2368
2369   handleBatch2Success() {
2370     this.setState({ batch2Correct: true });
2371   }
2372
2373   handleSuspectCorrect(isCorrect) {
2374     this.setState({ suspectCorrect: isCorrect });
2375   }
2376
2377   handleUserIdChange(e) {
2378     e.preventDefault();
2379     this.setState({ user_id: e.target.value });
2380   }
2381
2382   handlePasswordChange(e) {
```

```

2382     e.preventDefault()
2383     this.setState({ password: e.target.value });
2384 }
2385
2386 processResults1(results) {
2387     this.setState({ results1: results });
2388 }
2389
2390 processResults2(results) {
2391     this.setState({ results2: results });
2392 }
2393
2394 async handleQuery() {
2395     var response = await this.executeQuery(this.state.user_id,
2396     this.state.password);
2397     var isQuerySuccessful = response.isLoginSuccessful == 'true' ? true :
2398     false;
2399     this.setState({ isLoginSuccessful: isQuerySuccessful });
2400     this.setState({ loginClicked: true });
2401     this.setState({ errorMessage: response.error });
2402 }
2403
2404 async executeQuery(user_id, pwd) {
2405     const response = await fetch(BACKEND_API_URL + "/login", {
2406     method: "POST",
2407     mode: 'cors',
2408     headers: {
2409         'Content-Type': 'application/json'
2410     },
2411     body: JSON.stringify({
2412         user_id: user_id,
2413         password: pwd,
2414     })
2415     })
2416     return await response.json();
2417 }
2418
2419 render() {
2420     let suspect, table1 = null;
2421     if (this.state.batch1Correct) {
2422         suspect = <Container>
2423             <h5>Prime Suspect</h5>
2424             <p> Based on what we know about Tony's Questionnaire
2425                 information, where he was found when he died, and the
2426                 Purchase Order information, can you deduce who the prime
2427                 suspect should be?</p>
2428             <h6 className='sub-headers'> Declare Prime Suspect</h6>
2429             <p className="helper-text"><b>Use all the information you have
2430                 collected so far, including the files downloaded on your

```

```

    computer, to deduce who the prime suspect should be.</b></p>
2425     <Suspect game_step='S7_S' suspectCorrect=
        {this.handleSuspectCorrect}></Suspect>
2426     </Container>;
2427     table1 = <ResponseTable results={this.state.results1} />;
2428 }
2429 let injection2 = null;
2430 if (this.state.suspectCorrect) {
2431     injection2 = <Container>
2432         <h5>Get Suspect Credentials</h5>
2433         <p>You have Thanos' User_ID. Now all you have to do is find
            his password and log into his account to see if your hunch
            is correct about him being guilty!
2434         Since this database is vulnerable to SQL injection, it's
            likely that passwords aren't stored securely either.
2435         </p>
2436         <h6 className='sub-headers'> SQL Injection 2</h6>
2437         <p className='helper-text'><b>Use Batch SQL Injection to find
            out what Thanos's Password is.</b></p>
2438         <Hint hint="Use the USERS table to find Thanos' password."></
            Hint>
2439         <LoginSQL game_step='S7_B2' processResults=
            {this.processResults2} batchSqlCorrect=
            {this.handleBatch2Success} congratsMessage="Congratulations,
            your SQL Injection was successful! Here are the results of
            your query:" failureMessage="Hmm it doesn't look like your
            Injection Query was successful. Please try again."></
            LoginSQL>
2440     </Container>;
2441 }
2442 let queryResponse = null;
2443 if (this.state.loginClicked && !this.state.isLoginSuccessful) {
2444     queryResponse = <div className="instruction-div">
2445         <p>
2446             {this.state.errorMessage}
2447         </p>
2448     </div>;
2449 } else if (this.state.loginClicked && this.state.isLoginSuccessful) {
2450     queryResponse = <Container fluid="md">
2451         <Modal
2452             show={true}
2453             backdrop="static"
2454             keyboard={false}>
2455             <Modal.Dialog>
2456                 <Modal.Title>Congratulations!! You solved it!</
                    Modal.Title>
2457                 <Modal.Body>
2458                     <p>You solved Tony Stark's murder with your expert
                        SQL Injection skills!!<br><br><br></p>

```

```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 63
2459         Thanos was ready to take over Stark Industries  ↗
        (and the greater multiverse at large)
2460         so he broke into Pepper's computer and stole  ↗
        Tony's Questionnaire data. When he found out that Tony had  ↗
        an Almond allergy,
2461         he bought almond coffee creamer and swapped the  ↗
        label with a normal creamer. Since Tony's favorite drink  ↗
        is coffee and he's been working
2462         late nights, he drank a ton of coffee and had a  ↗
        severe allergic reaction to the creamer.
2463         <br></br><br></br>
2464         You have cleared your name and helped to avenge  ↗
        Tony's death!
2465         </p>
2466         <Button variant='primary' href='/'> End Game</  ↗
        Button>
2467         </Modal.Body>
2468         </Modal.Dialog>
2469     </Modal>
2470
2471     </Container>
2472 }
2473 let login, table2 = null;
2474 if (this.state.batch2Correct) {
2475     table2 = <ResponseTable results={this.state.results2} />;
2476     login = <Container>
2477         <h5>Final Step</h5>
2478         <p className='helper-text'> This is it! All of your evidence  ↗
        has seemed to converge on one prime suspect. Now it's time  ↗
        to see if all of your hard work has paid off! </p>
2479         <h6 className='sub-headers'>Login</h6>
2480         <p className='helper-text'><b>Enter your prime suspect's  ↗
        credentials to see if you've solved the murder! </b></p>
2481     <Container>
2482         <Row className="justify-content-md-center">
2483             <Col xs={8}>
2484                 <Form>
2485                     <div align='center' className='login-form'>
2486                         <h3 className='sub-headers'>Login</h3>
2487                         <Form.Group controlId='username'>
2488                             <Form.Label className='login-  ↗
        labels'>User_ID</Form.Label>
2489                             <Form.Control value=  ↗
        {this.state.user_id} type='username' placeholder="Enter  ↗
        User_ID here" onChange={this.handleUserIdChange}></  ↗
        Form.Control>
2490                     </Form.Group>
2491                     <Form.Group controlId='password'>
2492                         <Form.Label className='login-labels'  ↗

```

```

    >Password</Form.Label>
2493         <Form.Control value=
           {this.state.password} type= 'password' placeholder="Enter
           password here" onChange={this.handlePasswordChange} ></
           Form.Control>
2494         </Form.Group>
2495         <Button className='login-button'
           variant='primary' onClick={this.handleQuery} href=
           {this.props.history}>Login</Button>
           {queryResponse}
2496         </div>
2497       </Form>
2498     </Col>
2499   </Row>
2500 </Container>
2501 </Container>
2502 }
2503
2504 return (
2505   <Container fluid="md">
2506     <h2 className='sub-headers'>Step Seven: Targeted SQL and Final
2507       Conclusion</h2>
2508     <h5>Background</h5>
2509     <p>
2510       It looks like Peter Parker and Thanos are your new
2511       suspects. Either of them would've had time to set up their
2512       plan for Tony's demise
2513       before he arrived at the office at 11:30am. Now you just
2514       need to find out the substance that was used to murder
2515       Tony and who was responsible for using it.
2516       For any communal resources, employees must submit purchase
2517       orders to the company. Maybe someone slipped up and
2518       ordered their murder substance through the purchase order
2519       sheet?
2520     </p>
2521     <h5>PURCHASE_ORDER Table Information</h5>
2522     <p>
2523       Since you don't have any additional information about the
2524       PURCHASE_ORDERS table, it might be best to return as much
2525       information as possible from the table for your two
2526       suspects.
2527     </p>
2528     <h6 className='sub-headers'>SQL Injection 1</h6>
2529     <p className='helper-text'><b>Use Batch SQL Injection to
2530       return all columns and records for your suspects from the
2531       PURCHASE_ORDERS table.</b></p>
2532     <Hint hint={"Use any valid statement to finish the expected
2533       query. Use a SELECT statement with the PRAGMA function to
2534       find the column names. "}></Hint>

```



```

...sql-murder-mystery\sql-murder-mystery\public\Original.js 65
2521     <LoginSQL game_step='S7_B1' processResults=
        {this.processResults1} batchSqlCorrect=
        {this.handleBatch1Success} congratsMessage="Congratulations,
        your SQL Injection was successful! Here are the results of
        your query:" failureMessage="Hmm it doesn't look like your
        Injection Query was successful. Please try again." ></
        LoginSQL>
2522     {table1}
2523     {suspect}
2524     {injection2}
2525     {table2}
2526     {login}
2527     <Button variant="outline-primary float-left" href="/step6"
        >Back</Button>
2528   </Container>
2529   );
2530 }
2531 }
2532
2533 export default Step7;
2534
2535 //suspect.js
2536 // Written by Lia Ferguson
2537
2538 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
2539
2540 class Suspect extends React.Component {
2541   constructor(props) {
2542     super(props);
2543
2544     this.state = { isSuspect: false, isClicked: false, name: '',
        responseMessage: '' };
2545
2546     this.handleNameChange = this.handleNameChange.bind(this);
2547     this.handleSubmit = this.handleSubmit.bind(this);
2548   }
2549
2550   handleNameChange(e) {
2551     e.preventDefault();
2552     this.setState({ name: e.target.value });
2553   }
2554
2555   async handleSubmit() {
2556     var response = await this.checkName(this.state.name);
2557     var isQuerySuccessful = response.correct == 'true' ? true : false
2558     this.setState({ isClicked: true });
2559     this.setState({ isSuspect: isQuerySuccessful });
2560     this.props.suspectCorrect(this.state.isSuspect);
2561     this.setState({ responseMessage: response.message });

```

```

2562     }
2563
2564     async checkName(name, game_step) {
2565         const response = await fetch(BACKEND_API_URL + "/suspect", {
2566             method: "POST",
2567             mode: 'cors',
2568             headers: {
2569                 'Content-Type': 'application/json'
2570             },
2571             body: JSON.stringify({
2572                 name: name,
2573                 game_step: this.props.game_step
2574             })
2575         })
2576         return await response.json();
2577     }
2578
2579     render() {
2580         let validation = this.state.isClicked ? <Row><Col><p>
2581             {this.state.responseMessage}</p></Col></Row> : null;
2582         return (
2583             <Container fluid="md">
2584                 <Form>
2585                     <div align='center' className='login-form'>
2586                         <h6 className='sub-headers'>Enter Suspect</h6>
2587                         <Form.Row controlId='suspect'>
2588                             <Col >
2589                                 <Form.Label>Suspect Name</Form.Label>
2590                                 <Form.Control className='suspect-field' value=
2591                                     {this.state.name} placeholder="Enter suspect name here"
2592                                     onChange={this.handleNameChange}></Form.Control>
2593                                 </Col>
2594                                 <Col>
2595                                     <div className='button-padding'>
2596                                         <Button className='login-button'
2597                                             variant='primary' onClick={this.handleSubmit}> Submit</
2598                                             Button>
2599                                     </div>
2600                                 </Col>
2601                             </Form.Row>
2602                             {validation}
2603                         </div>
2604                     </Form>
2605                 </Container>
2606             );
2607     }
2608 }
2609
2610 export default Suspect;

```

```
2606
2607
2608 //trojanmodal.js
2609 // Written by Lia Ferguson
2610 const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';
2611
2612 class TrojanModal extends React.Component {
2613   constructor(props) {
2614     super(props);
2615
2616     this.state = { show: this.props.show, submitClicked: false,
2617                     isQuerySuccessful: false, f_name: '', l_name: '', nextButton:
2618                     false };
2619
2620     this.handleFirstNameChange = this.handleFirstNameChange.bind(this);
2621     this.handleLastNameChange = this.handleLastNameChange.bind(this);
2622     this.handleName = this.handleName.bind(this);
2623     this.handleClose = this.handleClose.bind(this);
2624     this.showNextButton = this.showNextButton.bind(this);
2625   }
2626
2627   handleFirstNameChange(e) {
2628     e.preventDefault();
2629     this.setState({ f_name: e.target.value });
2630   }
2631
2632   handleLastNameChange(e) {
2633     e.preventDefault();
2634     this.setState({ l_name: e.target.value });
2635   }
2636
2637   handleClose() {
2638     this.setState({ show: false });
2639   }
2640
2641   showNextButton() {
2642     this.setState({ nextButton: true })
2643   }
2644
2645   async handleName() {
2646     var response = await this.executeQuery(this.state.f_name,
2647                                           this.state.l_name);
2648     var isQuerySuccessful = response.isSuccess == 'true' ? true : false
2649     this.setState({ isQuerySuccessful: isQuerySuccessful });
2650     this.setState({ submitClicked: true });
2651     this.props.setTimeout(this.showNextButton, 4000);
2652   }
2653
2654   async executeQuery(f_name, l_name) {
```

```

2652     const response = await fetch(BACKEND_API_URL + "/trojan_horse", {
2653         method: "POST",
2654         mode: 'cors',
2655         headers: {
2656             'Content-Type': 'application/json'
2657         },
2658         body: JSON.stringify({
2659             first_name: f_name,
2660             last_name: l_name
2661         })
2662     })
2663     return await response.json();
2664 }
2665
2666 render() {
2667     let submitButton = <Button className='login-button' variant='primary'
2668     onClick={this.handleName}>Submit</Button>;
2669     if (this.state.submitClicked) {
2670         submitButton = <Button variant="primary" >
2671             <Spinner
2672                 as="span"
2673                 animation="border"
2674                 size="sm"
2675                 role="status"
2676                 aria-hidden="true"
2677             />
2678             Loading...
2679         </Button>;
2680
2681     let successText = '';
2682     let nextButton = null;
2683     if (this.state.nextButton) {
2684         submitButton = <Button disabled variant='primary'>Success</
2685         Button>;
2686         successText = <p>Your information was received! On to the
2687         shortcut!</p>;
2688         nextButton = <Button variant='primary' href='/'
2689         paddedcell'>Continue</Button>;
2690     }
2691
2692     return (
2693         <Container fluid="md">
2694             <Modal
2695                 show={this.state.show}
2696                 onHide={this.handleClose}
2697                 backdrop="static"
2698                 keyboard={false}>
2699                 <Modal.Dialog>

```

```

2697         <Modal.Header closeButton>
2698             <Modal.Title>Shortcut!</Modal.Title>
2699         </Modal.Header>
2700         <Modal.Body>
2701             <p>You have shown mastery of SQL Injection so far!
2702             Your skills have unlocked
2703             a shortcut to valuable information that will help
2704             you solve Tony Stark's murder
2705             quicker! The shortcut information will be
2706             downloaded to your computer.
2707             Please enter your first and last name so you can
2708             gain access to the shortcut.
2709             </p>
2710             <Form>
2711                 <div align='center' className='login-form'>
2712                     <h3 className='sub-headers'>Unlock
2713                     Shortcut</h3>
2714                     <Form.Group controlId='First Name'>
2715                         <Form.Label className='login-
2716                         labels'>First Name</Form.Label>
2717                         <Form.Control value=
2718                         {this.state.f_name} placeholder="Enter your first name
2719                         here" onChange={this.handleFirstNameChange}></
2720                         Form.Control>
2721                     </Form.Group>
2722                     <Form.Group controlId='password'>
2723                         <Form.Label className='login-labels'
2724                         >Last Name</Form.Label>
2725                         <Form.Control value=
2726                         {this.state.l_name} placeholder="Enter your last name
2727                         here" onChange={this.handleLastNameChange} ></
2728                         Form.Control>
2729                     </Form.Group>
2730                     {submitButton}
2731                     {successText}
2732                     {nextButton}
2733                 </div>
2734             </Form>
2735         </Modal.Body>
2736     </Modal.Dialog>
2737 </Modal>
2738 </Container>
2739 );
2740 }
2741 }
2742 export default ReactTimeout(TrojanModal);

```