```
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
                        been found dead in the office break room right next to
                       the coffee machine.
14                     Unfortunately, the police don’t have any suspects and the
                       case has gone cold.
15                     However, you think that one of your coworkers must be
                       responsible.
16                     In order to see who's had access to the breakroom, you
                       know you'll have to break into the secret employee
                       database on Tony's computer.
17                     Now, it’s up to you to use your SQL skills and find out
                       who murdered Tony Stark.
18         </p>
19                 <Row className="justify-content-md-center">
20                     <Col>
21                         <Button variant="outline-primary float-right" href="/
                       rules">Next</Button>
22                     </Col>
23                 </Row>
24             </Container>
25
26         );
27     }
28 }
29 export default Backstory;
30
31 //confirmsuspect.js
32 //Lia Ferguson
33 class ConfirmSuspect extends React.Component {
34     constructor(props) {
35         super(props);
36
37         this.state = { isClicked: false, isCorrect: false };
38         this.handleCorrectSuspectButton = this.handleCorrectSuspectButton.bind
             (this);
39     }
40
```

```
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>
```

```
            </Container>
        );
    }
}

export default ConfirmSuspect;

//Header.js
//Lia Ferguson and Julia Workum
class Header extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
        return (
            <Container fluid='true'>
                <Jumbotron>
                    <h1 class="display-4 header-content">Welcome to SQL Murder Mystery!</h1>
                    <p class="lead header-content">Can you figure out who murdered Tony Stark?</p>
                </Jumbotron>
            </Container>
        );
    }
}

export default Header;

//hint.js
// Written by Lia Ferguson

class Hint extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
        return (
            <Container>
                <Row className="justify-content-md-center">
                    <Col xs={8}>
                        <Accordion className='hint'>
                            <Card>
                                <Accordion.Toggle as={Card.Header} eventKey="0">
                                    Hint
                </Accordion.Toggle>
```

```
                                <Accordion.Collapse eventKey="0">
                                    <Card.Body>{this.props.hint}</Card.Body>
                                </Accordion.Collapse>
                            </Card>
                        </Accordion>
                    </Col>
                </Row>
            </Container>
        );
    }
}

export default Hint;

//loginsql.js
// lines 40, 48-50 written by Andrew Fecher
// all other code written by Lia Ferguson
const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';

class LoginSQL extends React.Component {
    constructor(props) {
        super(props);

        this.state = { isClicked: false, isQuerySuccessful: false, user_id:
          '', password: '', results: '', correctResults: false, errorMessage:
          '' };

        this.handleQuery = this.handleQuery.bind(this);
        this.handleUserIdChange = this.handleUserIdChange.bind(this);
        this.handlePasswordChange = this.handlePasswordChange.bind(this);
    }

    handleUserIdChange(e) {
        e.preventDefault()
        this.setState({ user_id: e.target.value });
    }

    handlePasswordChange(e) {
        e.preventDefault()
        this.setState({ password: e.target.value });
    }

    async handleQuery() {
        var response = await this.executeQuery(this.state.user_id,
          this.state.password);
        var isQuerySuccessful = response.isQuerySuccessful == 'true' ? true :
          false;
        var correctResults = response.correctResults == 'true' ? true : false;
        this.setState({ isQuerySuccessful: isQuerySuccessful });
```

```
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 &&
              this.state.correctResults) {
204             queryResponse = <div className="instruction-div">
205                 <p className="helper-text">
206                     {this.props.congratsMessage}
207                 </p>
208             </div>;
209             continueButton = <Button variant="outline-primary float-right"
                  href="/step2">Continue</Button>;
210         } else if (this.state.isClicked && (!this.state.isQuerySuccessful || !
              this.state.correctResults)) {
211             queryResponse = <div className="instruction-div">
212                 <p>
213                     {this.state.errorMessage}
214                 </p>
215             </div>;
216             continueButton = null;
```

```
        }
        return (
            <Container>
                <Row className="justify-content-md-center">
                    <Col xs={8}>
                        <Form>
                            <div align='center' className='login-form'>
                                <h3 className='sub-headers'>Login</h3>
                                <Form.Group controlId='username'>
                                    <Form.Label className='login-labels'>User_ID</Form.Label>
                                    <Form.Control value={this.state.user_id} type='username' placeholder="Enter User_ID here" onChange={this.handleUserIdChange}></Form.Control>
                                </Form.Group>
                                <Form.Group controlId='password'>
                                    <Form.Label className='login-labels'>Password</Form.Label>
                                    <Form.Control value={this.state.password} type='password' placeholder="Enter password here" onChange={this.handlePasswordChange} ></Form.Control>
                                </Form.Group>
                                <Button className='login-button' variant='primary' onClick={this.handleQuery}>Login</Button>
                                {queryResponse}
                            </div>
                        </Form>
                    </Col>
                </Row>
            </Container>
        );
    }
}

export default LoginSQL;

//paddedcell.js
// Written by Lia Ferguson
class PaddedCell extends React.Component {
    constructor(props) {
        super(props);

        this.state = { buttonClicked: false, correctQQ1: false, correctQQ2: false, correctQQ3: false };
        this.handleButtonClick = this.handleButtonClick.bind(this);
        this.handleCorrectChoice = this.handleCorrectChoice.bind(this);
    }

```

```
257     handleButtonClick() {
258         this.setState({ buttonClicked: true });
259     }
260
261
262     handleCorrectChoice(e) {
263         switch (e.target.value) {
264             case 'qq1':
265                 this.setState({ correctQQ1: true });
266                 break;
267             case 'qq2':
268                 this.setState({ correctQQ2: true });
269                 break;
270             case 'qq3':
271                 this.setState({ correctQQ3: true });
272                 break;
273         };
274     }
275
276     render() {
277         const qq1Question = 'The fake confession was downloaded onto your
              computer while it seemed like you were just submitting a shortcut
              form in the game. This action of executing a process behind the
              front of a “normal” looking process is what kind of attack?' ;
278
279         const qq1Answers = [
280             <Card.Text className="answer-text"> Memory Scanner <br></br><br></
                  br></Card.Text>,
281             <Card.Text className="answer-text"> Key Logger <br></br><br></
                  br></Card.Text>,
282             <Card.Text className="answer-text"> Trojan Horse <br></br><br></
                  br></Card.Text>,
283             <Card.Text className="answer-text"> Polymorphic Virus <br></
                  br><br></br></Card.Text>
284         ];
285
286         const qq1Header = new Map();
287         qq1Header['A'] = 'Incorrect';
288         qq1Header['B'] = 'Incorrect';
289         qq1Header['C'] = 'Correct';
290         qq1Header['D'] = 'Incorrect';
291
292         const qq1Responses = new Map();
293         qq1Responses['A'] = 'A memory scanner attack takes advantage of how
              memory is used on the host device. Nothing related to memory
              happened during this attack.';
294         qq1Responses['B'] = 'A key logger tracks your key strokes. While the
              window did collect your information, it was done via a form and not
              with key logging.';
```

```
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
```

```
            system, but it\'s still considered to be on the "right" side of the law.';
326         qq3Responses['B'] = 'Entrapment is an illegal practice that involves luring an attacker into a situation where they will break a law and be open to conviction.';
327
328         let question1, question2, question3 = null;
329         if (this.state.buttonClicked) {
330             question1 = <Container>
331                 <Row className="justify-content-md-center">
332                     <p className="q-text">
333                         <b>Question</b><br></br>
334                         {qq1Question}
335                     </p>
336                 </Row>
337                 <Row className="justify-content-md-center">
338                     <CardGroup>
339                         <Card style={{ width: '18rem' }} bg="light">
340                             <OverlayTrigger
341                                 trigger="click"
342                                 key='top'
343                                 rootClose
344                                 placement='top'
345                                 overlay={
346                                     <Popover id='popover-positioned-top'>
347                                         <Popover.Title as="h3">{qq1Header['A']}</Popover.Title>
348                                         <Popover.Content>
349                                             {qq1Responses['A']}
350                                         </Popover.Content>
351                                     </Popover>
352                                 }>
353                                 <Button className="question-button" variant="outline-primary" size="sm"> A </Button>
354                             </OverlayTrigger>
355                             <Card.Body className='question-text'>{qq1Answers[0]}</Card.Body>
356                         </Card>
357                         <Card style={{ width: '18rem' }} bg="light">
358                             <OverlayTrigger
359                                 trigger="click"
360                                 key='top'
361                                 rootClose
362                                 placement='top'
363                                 overlay={
364                                     <Popover id='popover-positioned-top'>
365                                         <Popover.Title as="h3">{qq1Header['B']}</Popover.Title>
366                                         <Popover.Content>
```

```
                                        {qq1Responses['B']}
                                    </Popover.Content>
                                </Popover>
                            }>
                            <Button className="question-button"
                        variant="outline-primary" size="sm" > B </Button>
                        </OverlayTrigger>
                        <Card.Body className='question-text'>{qq1Answers
                        [1]}</Card.Body>
                    </Card>
                </CardGroup>
            </Row>
            <Row className="justify-content-md-center">
                <CardGroup>
                    <Card style={{ width: '18rem' }} bg="light">
                        <OverlayTrigger
                            trigger="click"
                            key='bottom'
                            rootClose
                            placement='bottom'
                            overlay={
                                <Popover id='popover-positioned-bottom'>
                                    <Popover.Title as="h3">{qq1Header
                        ['C']}</Popover.Title>
                                    <Popover.Content>
                                        {qq1Responses['C']}
                                    </Popover.Content>
                                </Popover>
                            }>
                            <Button className="question-button"
                        variant="outline-primary" size="sm" value="qq1" onClick=
                        {this.handleCorrectChoice}> C </Button>
                        </OverlayTrigger>
                        <Card.Body className='question-text'>{qq1Answers
                        [2]}</Card.Body>
                    </Card>
                    <Card style={{ width: '18rem' }} bg="light">
                        <OverlayTrigger
                            trigger="click"
                            key='bottom'
                            rootClose
                            placement='bottom'
                            overlay={
                                <Popover id='popover-positioned-bottom'>
                                    <Popover.Title as="h3">{qq1Header
                        ['D']}</Popover.Title>
                                    <Popover.Content>
                                        {qq1Responses['D']}
                                    </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>
```

```
                    </CardGroup>
                </Row>
            </Container>;

            question3 = <Container>
                <Row className="justify-content-md-center">
                    <p className="q-text">
                        <b>Question</b><br></br>
                        {qq3Question}
                    </p>
                </Row>
                <Row className="justify-content-md-center">
                    <CardGroup>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='top'
                                rootClose
                                placement='top'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">{qq3Header
['A']}</Popover.Title>
                                        <Popover.Content>
                                            {qq3Responses['A']}
                                        </Popover.Content>
                                    </Popover>
                                }>
                                <Button className="question-button"
variant="outline-primary" size="sm"> A </Button>
                            </OverlayTrigger>
                            <Card.Body className='question-text'> {qq3Answers
[0]}</Card.Body>
                        </Card>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='top'
                                rootClose
                                placement='top'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">{qq3Header
['B']}</Popover.Title>
                                        <Popover.Content>
                                            {qq3Responses['B']}
                                        </Popover.Content>
                                    </Popover>
                                }>
```

```
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>
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.handleButtonClick}
                        > Begin </Button>
569                 </div>
570                 <br></br><br></br>
571                 {question1}
572                 <br></br><br></br>
573                 {question2}
574                 <br></br><br></br>
575                 {question3}
576                 <br></br><br></br>
577                 {sendoffMessage}
578                 {exitButton}
579 
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:
              '', sql2: '', sql2Response: '' }
593 
594         this.handleQuery = this.handleQuery.bind(this);
595         this.handleQueryChange = this.handleQueryChange.bind(this);
596         this.handleSQL1 = this.handleSQL1.bind(this);
597         this.handleSQL1Change = this.handleSQL1Change.bind(this);
598         this.handleSQL2 = this.handleSQL2.bind(this);
599         this.handleSQL2Change = this.handleSQL2Change.bind(this);
600     }
601     handleQueryChange(e) {
602         e.preventDefault()
603         this.setState({ query: e.target.value });
604     }
605     handleQuery() {
606         if (this.state.query == 'SELECT Password FROM Users WHERE name =
              \'John Doe\'') {
607             this.setState({ queryResponse: 'This is an accurate query to get
                  John Doe' });
608         }
609         else {
610             this.setState({ queryResponse: 'try again' });
611         }
```

```
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 generallyfirst 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>
                            <p> {this.state.sql1Response} </p>
                        </div>
                    </Form>
                </Col>
            </Row>
            <p>SQL Batch Injection can be utilitzed 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.
            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.
             </p>
            <br />
            <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:
                    <b>"; SELECT Computer_Number FROM Computers WHERE Name = 'John Doe' --
                    </b>
            </p>
            <Row className="justify-content-md-center">
                <Col xs={8}>
                    <Form>
                        <div align='center' className='login-form'>
                            <h3 className='sub-headers'>Login</h3>
                            <Form.Group controlId='username'>
                                <Form.Label className='login-labels'>User_ID</Form.Label>
                                <Form.Control value={this.state.sql2} type='username' placeholder="Enter SQL Batch injection here" onChange={this.handleSQL2Change}></Form.Control>
                            </Form.Group>
                            <Button className='login-button' variant='primary' onClick={this.handleSQL2}>Login</Button>
                            <p> {this.state.sql2Response} </p>
                        </div>
                    </Form>
                </Col>
            </Row>
            <br />
            <br />
            <p> <b>Good luck with the Mystery!</b></p>
            <Button variant="outline-primary float-left" href="/rules" >Back</Button>
            <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
                    Injection can be used to exploit webpages in order to
                    retrieve confidential information from their underlying
                    database.
776                     You will also test your knowledge of SQL Injection and
                    other important security topics through quiz-style
                    questions and interactive activities.
777                     Each step of the game will require you to successfully
                    execute SQL statements or prove your knowledge about SQL
                    Injection to uncover clues that will help you to solve the
                     unsolved murder of Tony Stark.
778           There will be helpful hints along the way if you need help with the
                task at hand. <br></br> <br></br>
779
780           As part of the game, text files may be downloaded onto your computer
                 in order to relay important clues or information to you. They
                will be stored in a file called "SQL-Mystery-Game-Files" on your
                Desktop. <b> Good luck and happy hacking!</b>
781
782                 </p>
783                 <Button variant="outline-primary float-left" href="/" >Back</
                  Button>
784                 <Button variant="outline-primary float-right" href="/practice"
                     >Practice!</Button>
785             </Container>
786
787         );
788     }
789 }
790 export default Rules;
791
792 //SQLInput.js
793 //Julia Workum
794 class SQLInput extends React.Component {
795     constructor(props) {
796         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:
              '', password: '' };
832
833         this.handleQuery = this.handleQuery.bind(this);
834         this.handleUserIdChange = this.handleUserIdChange.bind(this);
835         this.handlePasswordChange = this.handlePasswordChange.bind(this);
836     }
837
838     handleUserIdChange(e) {
839         e.preventDefault()
840         this.setState({ user_id: e.target.value });
841     }
842
843     handlePasswordChange(e) {
844         e.preventDefault()
```

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

```
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
                    the company database. There's only one problem: you do not
                     know his password to access the database. You can only
                    think of one option to get access to his computer, SQL
                    Injection! While it is a form of hacking, you deem it
                    worthy in order to try to find out who murdered your
                    friend, Tony.<br></br><br></br><b>Use SQL injection to
                    bypass the authentication system.</b>
895                 </p>
896                 <Row className="justify-content-md-center">
897                     <Col xs={8}>
898                         <Accordion className='hint'>
899                             <Card>
900                                 <Accordion.Toggle as={Card.Header}
                    eventKey="0">
901                                     Hint
902                 </Accordion.Toggle>
903                                 <Accordion.Collapse eventKey="0">
904                                     <Card.Body>Think back to the practice
                    section you just completed. Rely on you knowledge of how
                    to write comments in SQL, and where the admin account is
                    usually stored in a database to write a query that will
                    bypass the login function. </Card.Body>
905                                 </Accordion.Collapse>
906                             </Card>
907                         </Accordion>
908                     </Col>
909                 </Row>
910                 <Row className="justify-content-md-center">
911                     <Col xs={8}>
912                         <Form>
913                             <div align='center' className='login-form'>
914                                 <h3 className='sub-headers'>Login</h3>
915                                 <Form.Group controlId='username'>
916                                     <Form.Label className='login-
                    labels'>User_ID</Form.Label>
917                                     <Form.Control value={this.state.user_id}
                    type='username' placeholder="Enter User_ID here" onChange=
                    {this.handleUserIdChange}></Form.Control>
918                                 </Form.Group>
919                                 <Form.Group controlId='password'>
```

```
                                    <Form.Label className='login-labels'
                    >Password</Form.Label>
                                    <Form.Control value={this.state.password}
                    type='password' placeholder="Enter password here"
                    onChange={this.handlePasswordChange} ></Form.Control>
                                </Form.Group>
                                <Button className='login-button'
                    variant='primary' onClick={this.handleQuery}>Login</
                    Button>
                            </div>
                            {queryResponse}
                            {/*<div className='sql-results'>
                <ListGroup>
                  <ListGroup.Item> Results of SQL Injection</ListGroup.Item>
                  <ListGroup.Item>UserId = ...</ListGroup.Item>
                  <ListGroup.Item>Username = ...</ListGroup.Item>
                  <ListGroup.Item>Password = ...</ListGroup.Item>
                </ListGroup>
              </div>*/
                            }
                        </Form>
                    </Col>
                </Row>
                <Row className="justify-content-md-center">
                    <Col xs={6}>
                        <Button variant="outline-primary float-left" href="/
                    practice">Back</Button>
                        {continueButton}
                    </Col>
                </Row>
            </Container>

        );
    }
}

export default Step1;

//step2.js
//Julia Workum
class Step2 extends React.Component {
    constructor(props) {
        super(props);
        this.state = { q1Correct: false, q2Correct: false, q3Correct: false,
          q4Correct: false, q5Correct: false, minutes: 2, seconds: 0 };
        this.handleCorrectChoice = this.handleCorrectChoice.bind(this)
    }

    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
                        ['A']}</Popover.Title>
1121                                        <Popover.Content>
1122                                            {questionResponses1[ 'A']}
1123                                        </Popover.Content>
1124                                    </Popover>
1125                                }>
1126                                <Button className="question-button"
                            variant="outline-primary" size="sm"> A </Button>
1127                            </OverlayTrigger>
1128                            {questionAnswers1[0]}
1129                        </Card>
1130                        <Card style={{ width: '18rem' }} bg="light">
1131                            <OverlayTrigger
1132                                trigger="click"
1133                                key='top'
1134                                rootClose
1135                                placement='top'
1136                                overlay={
1137                                    <Popover id='popover-positioned-top'>
1138                                        <Popover.Title as="h3">{questionHeaders1
                        ['B']}</Popover.Title>
1139                                        <Popover.Content>
1140                                            {questionResponses1[ 'B']}
1141                                        </Popover.Content>
1142                                    </Popover>
1143                                }>
1144                                <Button className="question-button"
                            variant="outline-primary" size="sm"> B </Button>
1145                            </OverlayTrigger>
1146                            {questionAnswers1[1]}
1147                        </Card>
1148                    </CardGroup>
1149                </Row>
1150                <Row className="justify-content-md-center">
1151                    <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
                    ['C']}</Popover.Title>
1161                                    <Popover.Content>
1162                                        {questionResponses1[ 'C']}
1163                                    </Popover.Content>
1164                                </Popover>
1165                            }>
1166                            <Button className="question-button"
                    variant="outline-primary" size="sm"> C </Button>
1167                        </OverlayTrigger>
1168                        {questionAnswers1[2]}
1169                    </Card>
1170                    <Card style={{ width: '18rem' }} bg="light">
1171                        <OverlayTrigger
1172                            trigger="click"
1173                            key='bottom'
1174                            rootClose
1175                            placement='bottom'
1176                            overlay={
1177                                <Popover id='popover-positioned-top'>
1178                                    <Popover.Title as="h3">{questionHeaders1
                    ['D']}</Popover.Title>
1179                                    <Popover.Content>
1180                                        {questionResponses1[ 'D']}
1181                                    </Popover.Content>
1182                                </Popover>
1183                            }>
1184                            <Button className="question-button"
                    variant="outline-primary" size="sm" value='correct1'
                    onClick={this.handleCorrectChoice}> D </Button>
1185                        </OverlayTrigger>
1186                        {questionAnswers1[3]}
1187                    </Card>
1188                </CardGroup>
1189            </Row>
1190        </Container>;
1191
1192        let startOverButton = null;
1193        let timer = <h1>Time remaining: {minutes}:{seconds < 10 ? `0${seconds}
              ` : seconds} </h1>
1194        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!
                    </h1></Row>
1205                    <Row className="justify-content-md-center"><Button
                    variant="danger float-right" href="/step1">Start Over</
                    Button></Row>
1206                </Container>
1207        } else if (minutes === 0 && seconds === 0 && this.state.q5Correct) {
1208            timer = <Row className="justify-content-md-center"><h1>Success!</
                  h1></Row>
1209        }
1210
1211        let questionSetup2 = null;
1212        if (this.state.q1Correct) {
1213            questionSetup2 =
1214                <Container>
1215                    <Row className="justify-content-md-center">
1216                        {question2}
1217                    </Row>
1218                    <Row className="justify-content-md-center">
1219                        <CardGroup>
1220                            <Card style={{ width: '18rem' }} bg="light">
1221                                <OverlayTrigger
1222                                    trigger="click"
1223                                    key='top'
1224                                    rootClose
1225                                    placement='top'
1226                                    overlay={
1227                                        <Popover id='popover-positioned-top'>
1228                                            <Popover.Title as="h3">
                    {questionHeaders2['A']}</Popover.Title>
1229                                            <Popover.Content>
1230                                                {questionResponses2['A']}
1231                                            </Popover.Content>
1232                                        </Popover>
1233                                    }>
1234                                    <Button className="question-button"
                    variant="outline-primary" size="sm"> A </Button>
1235                                </OverlayTrigger>
1236                                {questionAnswers2[0]}
1237                            </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">{questionHeaders2['B']}</Popover.Title>
1247                                            <Popover.Content>
1248                                                {questionResponses2[ 'B']}
1249                                            </Popover.Content>
1250                                        </Popover>
1251                                    }>
1252                                    <Button className="question-button" variant="outline-primary" size="sm" value='correct2' onClick={this.handleCorrectChoice}> B </Button>
1253                                </OverlayTrigger>
1254                                {questionAnswers2[1]}
1255                            </Card>
1256                        </CardGroup>
1257                    </Row>
1258                    <Row className="justify-content-md-center">
1259                        <CardGroup>
1260                            <Card style={{ width: '18rem' }} bg="light">
1261                                <OverlayTrigger
1262                                    trigger="click"
1263                                    key='bottom'
1264                                    rootClose
1265                                    placement='bottom'
1266                                    overlay={
1267                                        <Popover id='popover-positioned-bottom'>
1268                                            <Popover.Title as="h3">{questionHeaders2['C']}</Popover.Title>
1269                                            <Popover.Content>
1270                                                {questionResponses2[ 'C']}
1271                                            </Popover.Content>
1272                                        </Popover>
1273                                    }>
1274                                    <Button className="question-button" variant="outline-primary" size="sm"> C </Button>
1275                                </OverlayTrigger>
1276                                {questionAnswers2[2]}
1277                            </Card>
1278                            <Card style={{ width: '18rem' }} bg="light">
1279                                <OverlayTrigger
1280                                    trigger="click"
```

```
                                    key='bottom'
                                    rootClose
                                    placement='bottom'
                                    overlay={
                                        <Popover id='popover-positioned-top'>
                                            <Popover.Title as="h3">
                    {questionHeaders2['D']}</Popover.Title>
                                            <Popover.Content>
                                                {questionResponses2[ 'D']}
                                            </Popover.Content>
                                        </Popover>
                                    }>
                                    <Button className="question-button"
                    variant="outline-primary" size="sm"> D </Button>
                                </OverlayTrigger>
                                {questionAnswers2[3]}
                            </Card>
                        </CardGroup>
                    </Row>
                </Container>
        }

        let questionSetup3 = null;
        if (this.state.q2Correct) {
            questionSetup3 =
                <Container>
                    <Row className="justify-content-md-center">
                        {question3}
                    </Row>
                    <Row className="justify-content-md-center">
                        <CardGroup>
                            <Card style={{ width: '18rem' }} bg="light">
                                <OverlayTrigger
                                    trigger="click"
                                    key='top'
                                    rootClose
                                    placement='top'
                                    overlay={
                                        <Popover id='popover-positioned-top'>
                                            <Popover.Title as="h3">
                    {questionHeaders3['A']}</Popover.Title>
                                            <Popover.Content>
                                                {questionResponses3[ 'A']}
                                            </Popover.Content>
                                        </Popover>
                                    }>
                                    <Button className="question-button"
                    variant="outline-primary" size="sm" value='correct3'
                    onClick={this.handleCorrectChoice}> T </Button>
```

```
1325                                    </OverlayTrigger>
1326                                    {questionAnswers3[0]}
1327                                </Card>
1328                                <Card style={{ width: '18rem' }} bg="light">
1329                                    <OverlayTrigger
1330                                        trigger="click"
1331                                        key='top'
1332                                        rootClose
1333                                        placement='top'
1334                                        overlay={
1335                                            <Popover id='popover-positioned-top'>
1336                                                <Popover.Title as="h3">
                        {questionHeaders3['B']}</Popover.Title>
1337                                                <Popover.Content>
1338                                                    {questionResponses3[ 'B']}
1339                                                </Popover.Content>
1340                                            </Popover>
1341                                        }>
1342                                        <Button className="question-button"
                        variant="outline-primary" size="sm"> F </Button>
1343                                    </OverlayTrigger>
1344                                    {questionAnswers3[1]}
1345                                </Card>
1346                            </CardGroup>
1347                        </Row>
1348                    </Container>
1349            }
1350
1351            let questionSetup4 = null;
1352            if (this.state.q3Correct) {
1353                questionSetup4 =
1354                    <Container>
1355                        <Row className="justify-content-md-center">
1356                            {question4}
1357                        </Row>
1358                        <Row className="justify-content-md-center">
1359                            <CardGroup>
1360                                <Card style={{ width: '18rem' }} bg="light">
1361                                    <OverlayTrigger
1362                                        trigger="click"
1363                                        key='top'
1364                                        rootClose
1365                                        placement='top'
1366                                        overlay={
1367                                            <Popover id='popover-positioned-top'>
1368                                                <Popover.Title as="h3">
                        {questionHeaders4['A']}</Popover.Title>
1369                                                <Popover.Content>
1370                                                    {questionResponses4[ 'A']}
```

```
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>
                            </OverlayTrigger>
                            {questionAnswers4[2]}
                        </Card>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='bottom'
                                rootClose
                                placement='bottom'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">
                    {questionHeaders4['D']}</Popover.Title>
                                        <Popover.Content>
                                            {questionResponses4[ 'D']}
                                        </Popover.Content>
                                    </Popover>
                                }>
                                <Button className="question-button"
                    variant="outline-primary" size="sm"> D </Button>
                            </OverlayTrigger>
                            {questionAnswers4[3]}
                        </Card>
                    </CardGroup>
                </Row>
            </Container>
        }

        let questionSetup5 = null;
        if (this.state.q4Correct) {
            questionSetup5 =
                <Container>
                    <Row className="justify-content-md-center">
                        {question5}
                    </Row>
                    <Row className="justify-content-md-center">
                        <CardGroup>
                            <Card style={{ width: '18rem' }} bg="light">
                                <OverlayTrigger
                                    trigger="click"
                                    key='top'
                                    rootClose
                                    placement='top'
                                    overlay={
                                        <Popover id='popover-positioned-top'>
                                            <Popover.Title as="h3">
                    {questionHeaders5['A']}</Popover.Title>
```

```
                                        <Popover.Content>
                                            {questionResponses5[ 'A']}
                                        </Popover.Content>
                                    </Popover>
                                }>
                                <Button className="question-button" variant="outline-primary" size="sm"> A </Button>
                            </OverlayTrigger>
                            {questionAnswers5[0]}
                        </Card>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='top'
                                rootClose
                                placement='top'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">{questionHeaders5['B']}</Popover.Title>
                                        <Popover.Content>
                                            {questionResponses5[ 'B']}
                                        </Popover.Content>
                                    </Popover>
                                }>
                                <Button className="question-button" variant="outline-primary" size="sm" value='correct5' onClick={this.handleCorrectChoice}> B </Button>
                            </OverlayTrigger>
                            {questionAnswers5[1]}
                        </Card>
                    </CardGroup>
                </Row>
                <Row className="justify-content-md-center">
                    <CardGroup>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='bottom'
                                rootClose
                                placement='bottom'
                                overlay={
                                    <Popover id='popover-positioned-bottom'>
                                        <Popover.Title as="h3">{questionHeaders5['C']}</Popover.Title>
                                        <Popover.Content>
                                            {questionResponses5[ 'C']}
                                        </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,
                  nextButton = null;
1594            let loginSQL = <LoginSQL processResults={this.processResults}
                  game_step='S3_B1' batchSqlCorrect={this.handleBatchQuerySuccess}
                  congratsMessage="Congratulations, your SQL Injection was
                  successful!" failureMessage="Hmm it doesn't look like your Injection
                   Query was successful. Please try again."></LoginSQL>;
1595
1596            if (this.state.step == 1) {
1597                nextButton = <Button variant="outline-primary float-center"
                      onClick={this.advance}>Got it!</Button>;
1598            }
1599            if (this.state.step > 0) {
1600                queryResponse = <div>
1601                    <p> Logins do not directly output the results of thier queries
                           to the front end, however they can be manipulated to do so
                          using methods such as inducing errors, manipulating the
                          outfile, sniffing, among many others. <br />
1602                        Luckily many progams such as SQLMap and other will do this
                           automatically. <br />
1603                        The system being utilized here will do this automatically
                          and just return the value of the second query made.
1604                    </p>
1605                    {nextButton}
1606                </div>;
1607            }
1608            if (this.state.step > 1) {
1609                table = <ResponseTable results={this.state.results} />;
1610                continueButton = continueButton = <Button variant="outline-primary
                       float-right" href="/step4">Continue</Button>;
1611            }
1612
1613            return (
1614                <Container fluid='md'>
1615                    <h2 className='sub-headers'>Step Three: Figure out the table
                          names</h2>
1616                    <p>In order to retrieve information from the database, you
                          will need more information about the underlying database
                          schema. <b>Use SQL injection to retrieve the names of the
                          tables in the database.</b></p>
1617                    <Row className="justify-content-md-center">
1618                        <Col xs={8}>
1619                            <Container>
1620                                <h5>Table Names</h5>
```

```
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,
                  batchSql2Correct: false,
1650             results1: '', results2: ''
1651         };
1652
1653         this.handleClick = this.handleClick.bind(this);
1654         this.handleCorrectChoice = this.handleCorrectChoice.bind(this);
1655         this.handleBatchQuerySuccess = this.handleBatchQuerySuccess.bind
              (this);
1656         this.handleBatchQuery2Success = this.handleBatchQuery2Success.bind
              (this);
1657         this.processResults1 = this.processResults1.bind(this);
1658         this.processResults2 = this.processResults2.bind(this);
1659     }
1660
```

```
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
            from SQL Injection?';
1697          const userAnswers = [
1698              <Card.Text className="answer-text"> Encrypt all data stored in the
                database <br></br><br></br></Card.Text>,
1699              <Card.Text className="answer-text"> Limit admin access to as few
                users as possible <br></br><br></br></Card.Text>,
1700              <Card.Text className="answer-text"> Make a really complex schema
                that would be hard for an attacker to reverse engineer <br></
                br><br></br></Card.Text>,
1701              <Card.Text className="answer-text"> Sanitize all user input from
                form fields <br></br><br></br></Card.Text>
1702          ];
1703
```

```
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
1722              <br></br><br></br></Card.Text>,
1723              <Card.Text className="answer-text"> Use a packet sniffer to
                    examine packets coming in and out of network ports related to
                    the application
1724              <br></br><br></br></Card.Text>
1725          ];
1726
1727          const qq1Header = new Map();
1728          qq1Header['A'] = 'Incorrect';
1729          qq1Header['B'] = 'Correct';
1730          qq1Header['C'] = 'Incorrect';
1731          qq1Header['D'] = 'Incorrect';
1732
1733          const qq1Responses = new Map();
1734          qq1Responses['A'] = 'Even though values are being guessed, this is
                really just using the website as it was intended';
1735          qq1Responses['B'] = 'Internal server errors or generic error messages
```

```
            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>
```

```
                <p className="helper-text">
                    <b>Answer the following security question to reveal the column names you need to query from the USER_INFO table.</b>
                </p>
            </div>;

            userIdQuestion = <Container>
                <Row className="justify-content-md-center">
                    <p>
                        <b>Question</b><br></br>
                        {userQuestion}
                    </p>
                </Row>
                <Row className="justify-content-md-center">
                    <CardGroup>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='top'
                                rootClose
                                placement='top'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">{userHeader['A']}</Popover.Title>
                                        <Popover.Content>
                                            {userResponses['A']}
                                        </Popover.Content>
                                    </Popover>
                                }>
                                <Button className="question-button" variant="outline-primary" size="sm"> A </Button>
                            </OverlayTrigger>
                            {userAnswers[0]}
                        </Card>
                        <Card style={{ width: '18rem' }} bg="light">
                            <OverlayTrigger
                                trigger="click"
                                key='top'
                                rootClose
                                placement='top'
                                overlay={
                                    <Popover id='popover-positioned-top'>
                                        <Popover.Title as="h3">{userHeader['B']}</Popover.Title>
                                        <Popover.Content>
                                            {userResponses['B']}
                                        </Popover.Content>
```

```
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
                        columns you will need are: User_ID, First_name, Last_name</
                        p>
1868                    <h6 className="sub-headers"> SQL Injection</h6>
1869                    <p className="helper-text"><b>Use Batch SQL Injection to
                        retrieve all name and User ID records from the USER_INFO
                        table. Make special note of Tony's ID. </b></p>
1870                </div>;
1871                batchInjectSection = <div>
1872                    <Hint hint={"Batch injection is performed by completing the
                        expected query and ending it with a semi colon, and then
                        typing another query following it that will retrieve the
                        information you desire from the database."}></Hint>
1873                    <LoginSQL game_step='S4_B1' processResults=
                        {this.processResults1} batchSqlCorrect=
                        {this.handleBatchQuerySuccess}
                        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>
1874                </div>
1875                table1 = <ResponseTable results={this.state.results1} />;
1876            }
1877            let questionnaireBackground = null;
1878            let questionnaireQuestions = null;
1879
1880            if (this.state.batchSqlCorrect) {
1881                batchInjectFinal = <div className='text-under-table'>
1882                    <p className="helper-text">Well done! Now we know how to
                        identify Tony's information in all of the database tables.</
                        p>
1883                </div>;
1884                questionnaireBackground = <div>
1885                    <h5>Background</h5>
1886                    <p>
```

```
1887                     The police report also mentioned that Pepper Pots sent out
                          a questionnaire to all the company's employees
1888                     to collect personal information from employees in order to
                          tailor events to their food and activity preferences.
1889                     Pepper kept a copy of the questionnaire responses in a
                         text document in her company account in addition to the
                         database copy,
1890                     and police suspect that the murderer hacked into her
                         account to retrieve the personal information about Tony so
                          they
1891                     could plot their crime. <br></br><br></br>
1892                 </p>
1893                 <p className="helper-text">
1894                     <b>Answer the following security questions to reveal the
                         column names you need to query the QUESTIONNAIRE table for
                          Tony's responses.</b>
1895                 </p>
1896             </div>;
1897             questionnaireQuestions = <Container>
1898                 <Container>
1899                     <Row>
1900                         <p>
1901                             <b>Question</b><br></br>
1902                             {qq1Question}
1903                         </p>
1904                     </Row>
1905                     <Row >
1906                         <Col>
1907                             <CardGroup>
1908                                 <Card style={{ width: '18rem' }} bg="light">
1909                                     <OverlayTrigger
1910                                         trigger="click"
1911                                         key='top'
1912                                         rootClose
1913                                         placement='top'
1914                                         overlay={
1915                                             <Popover id='popover-positioned-
                    top'>
1916                                                 <Popover.Title as="h3">
                    {qq1Header['A']}</Popover.Title>
1917                                                 <Popover.Content>
1918                                                     {qq1Responses[ 'A']}
1919                                                 </Popover.Content>
1920                                             </Popover>
1921                                         }>
1922                                         <Button className="question-button"
                    variant="outline-primary" size="sm"> A </Button>
1923                                     </OverlayTrigger>
1924                                     {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' }}
                    bg="light">
2023                                            <OverlayTrigger
2024                                                trigger="click"
2025                                                key='top'
2026                                                rootClose
2027                                                placement='top'
2028                                                overlay={
2029                                                    <Popover id='popover-
                    positioned-top'>
2030                                                        <Popover.Title
                    as="h3">{qq2Header['A']}</Popover.Title>
2031                                                        <Popover.Content>
2032                                                            {qq2Responses
                    ['A']}
2033                                                        </Popover.Content>
2034                                                    </Popover>
2035                                                }>
2036                                                <Button className="question-
                    button" variant="outline-primary" size="sm" value="qq2"
                    onClick={this.handleCorrectChoice} > A </Button>
2037                                            </OverlayTrigger>
2038                                            {qq2Answers[0]}
2039                                        </Card>
2040                                        <Card style={{ width: '16rem' }}
                    bg="light">
2041                                            <OverlayTrigger
2042                                                trigger="click"
2043                                                key='top'
2044                                                rootClose
2045                                                placement='top'
2046                                                overlay={
2047                                                    <Popover id='popover-
                    positioned-top'>
2048                                                        <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
2099                      in the partial schema we retrieved.
2100                  </p>
2101                  <h5>Employee User ID</h5>
2102                  <p>
2103                      Everyone in the company has a unique user ID number that
                        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!</
                    Popover.Title>
2130                            </Popover>
2131                        }>
2132                        <Button variant='outline-primary' onClick=
                    {this.handleClick}>USER_INFO</Button>
2133                    </OverlayTrigger>
2134                    <OverlayTrigger
2135                        trigger="click"
2136                        key='top'
2137                        rootClose
2138                        placement='top'
2139                        overlay={
2140                            <Popover id='popover-positioned-top'>
2141                                <Popover.Title as="h3">Incorrect</
                    Popover.Title>
2142                                <Popover.Content>Trial and error when
                    navigating a schema blind is completely normal! Please try
                     again!</Popover.Content>
2143                            </Popover>
2144                        }>
2145                        <Button variant='outline-primary'>QUESTIONNAIRE</
                    Button>
2146                    </OverlayTrigger>
2147                    <OverlayTrigger
2148                        trigger="click"
2149                        key='top'
2150                        rootClose
2151                        placement='top'
2152                        overlay={
2153                            <Popover id='popover-positioned-top'>
2154                                <Popover.Title as="h3">Incorrect</
                    Popover.Title>
2155                                <Popover.Content>Trial and error when
                    navigating a schema blind is completely normal! Please try
                     again!</Popover.Content>
2156                            </Popover>
2157                        }>
2158                        <Button variant='outline-primary'>PURCHASE_ORDERS</
                    Button>
```

```
                    </OverlayTrigger>
                    <OverlayTrigger
                        trigger="click"
                        key='top'
                        rootClose
                        placement='top'
                        overlay={
                            <Popover id='popover-positioned-top'>
                                <Popover.Title as="h3">Incorrect</Popover.Title>
                                <Popover.Content>Trial and error when navigating a schema blind is completely normal! Please try again!</Popover.Content>
                            </Popover>
                        }>
                        <Button variant='outline-primary'>BUILDING_ACCESS</Button>
                    </OverlayTrigger>
                </ButtonGroup>
                <br />
                <br />
                <br />
                {userIdColumnText}
                {userIdQuestion}
                {batchInjectInstructions}
                {batchInjectSection}
                {table1}
                {batchInjectFinal}
                {questionnaireBackground}
                {questionnaireQuestions}
                {batchInjectInstructions2}
                {batchInjectSection2}
                {table2}
                {batchFileOutput}
                <Button variant="outline-primary float-left" href="/step3"
                  >Back</Button>
                {continueButton}
            </Container>
        );
    }
}
export default Step4;

//step6.js
//Written by Lia Ferguson
const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';

class Step6 extends React.Component {
    constructor(props) {
```

```
2203          super(props);
2204
2205          this.state = {
2206              batch1Correct: false, batch2Correct: false, batch3Correct: false,
                    confirmSuspectCorrect: false,
2207              suspect1Correct: false, suspect2Correct: false, results1: '',
                    results2: '', results3: ''
2208          };
2209
2210          this.handleBatch1Success = this.handleBatch1Success.bind(this);
2211          this.handleBatch2Success = this.handleBatch2Success.bind(this);
2212          this.handleBatch3Success = this.handleBatch3Success.bind(this);
2213          this.handleConfirmSuspect = this.handleConfirmSuspect.bind(this);
2214          this.handleSuspect1Correct = this.handleSuspect1Correct.bind(this);
2215          this.handleSuspect2Correct = this.handleSuspect2Correct.bind(this);
2216          this.processResults1 = this.processResults1.bind(this);
2217          this.processResults2 = this.processResults2.bind(this);
2218          this.processResults3 = this.processResults3.bind(this);
2219      }
2220
2221      handleBatch1Success() {
2222          this.setState({ batch1Correct: true });
2223      }
2224
2225      handleBatch2Success() {
2226          this.setState({ batch2Correct: true });
2227      }
2228
2229      handleBatch3Success() {
2230          this.setState({ batch3Correct: true });
2231      }
2232
2233      handleConfirmSuspect(isCorrect) {
2234          this.setState({ confirmSuspectCorrect: isCorrect })
2235      }
2236
2237      handleSuspect1Correct(isCorrect) {
2238          this.setState({ suspect1Correct: isCorrect })
2239      }
2240
2241      handleSuspect2Correct(isCorrect) {
2242          this.setState({ suspect2Correct: isCorrect })
2243      }
2244
2245      processResults1(results) {
2246          this.setState({ results1: results });
2247      }
2248
2249      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
                      Natasha and Bruce entered the building. Remember, you have
                      their User_ID's in your 'Clues.txt' file</p>
2263                  <h6 className='sub-headers'> SQL Injection 2</h6>
2264                  <p className='helper-text'><b>Use SQL Injection to find out
                      what time Natasha and Bruce accessed the building on the day
                       of Tony's death. Remember to return the User_ID column as
                      well so you know who entered the building when!</b></p>
2265                  <LoginSQL game_step='S6_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>
2266              </Container>;
2267          }
2268
2269          let table2 = null;
2270          if (this.state.batch2Correct) {
2271              table2 = <ResponseTable results={this.state.results2} />;
2272              confirmSuspect = <ConfirmSuspect suspectConfirmCorrect=
                    {this.handleConfirmSuspect}></ConfirmSuspect>;
2273          }
2274
2275          let injection3, table3 = null;
2276          if (this.state.confirmSuspectCorrect) {
2277              injection3 = <Container>
2278                  <h5>New Suspect Building Access</h5>
2279                  <p className='helper-text'>Hmm... it seems that both Natasha
                      or Bruce didn't check into the building until <b><i>after</
                      i></b> Tony was found in the break room.
2280                        Looks like the lead on liking almonds wasn't quite as
                      fruitful as we hoped. Let's pivot and find out who was in
                      the building before Tony to gain our next round of
                      suspects.</p>
2281                  <h6 className='sub-headers'> SQL Injection 3</h6>
```

```
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
```

```
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        }
```

```
}

export default Step6;

//step7.js
// Written by Lia Ferguson

const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';

class Step7 extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            batch1Correct: false, batch2Correct: false, user_id: '', password: ''
            isLoginSuccessful: false, errorMessage: '', loginClicked: false, suspectCorrect: false,
            results1: '', results2: ''
        };

        this.handleBatch1Success = this.handleBatch1Success.bind(this);
        this.handleBatch2Success = this.handleBatch2Success.bind(this);
        this.handleQuery = this.handleQuery.bind(this);
        this.handleSuspectCorrect = this.handleSuspectCorrect.bind(this);
        this.handleUserIdChange = this.handleUserIdChange.bind(this);
        this.handlePasswordChange = this.handlePasswordChange.bind(this);
        this.processResults1 = this.processResults1.bind(this);
        this.processResults2 = this.processResults2.bind(this);
    }

    handleBatch1Success() {
        this.setState({ batch1Correct: true });
    }

    handleBatch2Success() {
        this.setState({ batch2Correct: true });
    }

    handleSuspectCorrect(isCorrect) {
        this.setState({ suspectCorrect: isCorrect })
    }

    handleUserIdChange(e) {
        e.preventDefault()
        this.setState({ user_id: e.target.value });
    }

    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,
              this.state.password);
2396          var isQuerySuccessful = response.isLoginSuccessful == 'true' ? true :
              false;
2397          this.setState({ isLoginSuccessful: isQuerySuccessful });
2398          this.setState({ loginClicked: true });
2399          this.setState({ errorMessage: response.error });
2400      }
2401
2402      async executeQuery(user_id, pwd) {
2403          const response = await fetch(BACKEND_API_URL + "/login", {
2404              method: "POST",
2405              mode: 'cors',
2406              headers: {
2407                  'Content-Type': 'application/json'
2408              },
2409              body: JSON.stringify({
2410                  user_id: user_id,
2411                  password: pwd,
2412              })
2413          })
2414          return await response.json();
2415      }
2416
2417      render() {
2418          let suspect, table1 = null;
2419          if (this.state.batch1Correct) {
2420              suspect = <Container>
2421                  <h5>Prime Suspect</h5>
2422                  <p> Based on what we know about Tony's Questionnaire
                      information, where he was found when he died, and the
                      Purchase Order information, can you deduce who the prime
                      suspect should be?</p>
2423                  <h6 className='sub-headers'> Declare Prime Suspect</h6>
2424                  <p className="helper-text"><b>Use all the information you have
                        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></br>
```

```
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>
2496                                {queryResponse}
2497                            </div>
2498                        </Form>
2499                    </Col>
2500                </Row>
2501            </Container>
2502        </Container>
2503    }
2504
2505    return (
2506        <Container fluid="md">
2507            <h2 className='sub-headers'>Step Seven: Targeted SQL and Final
                   Conclusion</h2>
2508            <h5>Background</h5>
2509            <p>
2510                It looks like Peter Parker and Thanos are your new
                    suspects. Either of them would've had time to set up their
                     plan for Tony's demise
2511                    before he arrived at the office at 11:30am. Now you just
                    need to find out the substance that was used to murder
                    Tony and who was responsible for using it.
2512                    For any communal resources, employees must submit purchase
                     orders to the company. Maybe someone slipped up and
                    ordered their murder substance through the purchase order
                    sheet?
2513            </p>
2514            <h5>PURCHASE_ORDER Table Information</h5>
2515            <p>
2516                Since you don't have any additional information about the
                    PURCHASE_ORDERS table, it might be best to return as much
                    information as possible from the table for your two
                    suspects.
2517            </p>
2518            <h6 className='sub-headers'> SQL Injection 1</h6>
2519            <p className='helper-text'><b>Use Batch SQL Injection to
                  return all columns and records for your suspects from the
                  PURCHASE_ORDERS table.</b></p>
2520            <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>
```

```
                <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>
                {table1}
                {suspect}
                {injection2}
                {table2}
                {login}
                <Button variant="outline-primary float-left" href="/step6"
                  >Back</Button>
            </Container>
        );
    }
}

export default Step7;

//suspect.js
// Written by Lia Ferguson

const BACKEND_API_URL = 'http://127.0.0.1:5000/endpoints';

class Suspect extends React.Component {
    constructor(props) {
        super(props);

        this.state = { isSuspect: false, isClicked: false, name: '',
          responseMessage: '' };

        this.handleNameChange = this.handleNameChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
    }

    handleNameChange(e) {
        e.preventDefault();
        this.setState({ name: e.target.value });
    }

    async handleSubmit() {
        var response = await this.checkName(this.state.name);
        var isQuerySuccessful = response.correct == 'true' ? true : false
        this.setState({ isClicked: true });
        this.setState({ isSuspect: isQuerySuccessful });
        this.props.suspectCorrect(this.state.isSuspect);
        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>
              {this.state.responseMessage}</p></Col></Row> : null;
2581          return (
2582              <Container fluid="md">
2583                  <Form>
2584                      <div align='center' className='login-form'>
2585                          <h6 className='sub-headers'>Enter Suspect</h6>
2586                          <Form.Row controlId='suspect'>
2587                              <Col >
2588                                  <Form.Label>Suspect Name</Form.Label>
2589                                  <Form.Control className='suspect-field' value=
                      {this.state.name} placeholder="Enter suspect name here"
                      onChange={this.handleNameChange}></Form.Control>
2590                              </Col>
2591                              <Col>
2592                                  <div className='button-padding'>
2593                                      <Button className='login-button'
                      variant='primary' onClick={this.handleSubmit}> Submit</
                      Button>
2594                                  </div>
2595                              </Col>
2596                          </Form.Row>
2597                          {validation}
2598                      </div>
2599                  </Form>
2600              </Container>
2601          );
2602      }
2603  }
2604
2605  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,
              isQuerySuccessful: false, f_name: '', l_name: '', nextButton:
              false };
2617
2618         this.handleFirstNameChange = this.handleFirstNameChange.bind(this);
2619         this.handleLastNameChange = this.handleLastNameChange.bind(this);
2620         this.handleName = this.handleName.bind(this);
2621         this.handleClose = this.handleClose.bind(this);
2622         this.showNextButton = this.showNextButton.bind(this);
2623     }
2624
2625     handleFirstNameChange(e) {
2626         e.preventDefault();
2627         this.setState({ f_name: e.target.value });
2628     }
2629
2630     handleLastNameChange(e) {
2631         e.preventDefault();
2632         this.setState({ l_name: e.target.value });
2633     }
2634
2635     handleClose() {
2636         this.setState({ show: false });
2637     }
2638
2639     showNextButton() {
2640         this.setState({ nextButton: true })
2641     }
2642
2643     async handleName() {
2644         var response = await this.executeQuery(this.state.f_name,
              this.state.l_name);
2645         var isQuerySuccessful = response.isSuccess == 'true' ? true : false
2646         this.setState({ isQuerySuccessful: isQuerySuccessful });
2647         this.setState({ submitClicked: true });
2648         this.props.setTimeout(this.showNextButton, 4000);
2649     }
2650
2651     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'
              onClick={this.handleName}>Submit</Button>;
2668          if (this.state.submitClicked) {
2669              submitButton = <Button variant="primary" >
2670                  <Spinner
2671                      as="span"
2672                      animation="border"
2673                      size="sm"
2674                      role="status"
2675                      aria-hidden="true"
2676                  />
2677                  Loading...
2678              </Button>;
2679          }
2680
2681          let successText = '';
2682          let nextButton = null;
2683          if (this.state.nextButton) {
2684              submitButton = <Button disabled variant='primary'>Success</
                Button>;
2685              successText = <p>Your information was received! On to the
                shortcut!</p>;
2686              nextButton = <Button variant='primary' href='/
                paddedcell'>Continue</Button>;
2687          }
2688
2689          return (
2690              <Container fluid="md">
2691                  <Modal
2692                      show={this.state.show}
2693                      onHide={this.handleClose}
2694                      backdrop="static"
2695                      keyboard={false}>
2696                      <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!
                         Your skills have unlocked
2702                             a shortcut to valuable information that will help
                        you solve Tony Stark's murder
2703                             quicker! The shortcut information will be
                        downloaded to your computer.
2704                             Please enter your first and last name so you can
                        gain access to the shortcut.
2705                             </p>
2706                             <Form>
2707                                 <div align='center' className='login-form'>
2708                                     <h3 className='sub-headers'>Unlock
                        Shortcut</h3>
2709                                     <Form.Group controlId='First Name'>
2710                                         <Form.Label className='login-
                        labels'>First Name</Form.Label>
2711                                         <Form.Control value=
                        {this.state.f_name} placeholder="Enter your first name
                        here" onChange={this.handleFirstNameChange}></
                        Form.Control>
2712                                     </Form.Group>
2713                                     <Form.Group controlId='password'>
2714                                         <Form.Label className='login-labels'
                        >Last Name</Form.Label>
2715                                         <Form.Control value=
                        {this.state.l_name} placeholder="Enter your last name
                        here" onChange={this.handleLastNameChange} ></
                        Form.Control>
2716                                     </Form.Group>
2717                                     {submitButton}
2718                                     {successText}
2719                                     {nextButton}
2720                                 </div>
2721                             </Form>
2722                         </Modal.Body>
2723                     </Modal.Dialog>
2724                 </Modal>
2725
2726             </Container>
2727         );
2728     }
2729 }
2730
2731 export default ReactTimeout(TrojanModal);
```

```python
# from endpoints.py
""" endpoints written by Lia Ferguson:
        /login_bypass
        /login_query
        /trojan_horse
        /suspect
        /login

        with the exception of the two try - except blocks and json.dumps()
          lines written by Andrew Fecher
        in /login_query
"""
NUM_RECORDS_USERS_TABLE = 8
bp = Blueprint('endpoints', __name__, url_prefix='/endpoints')

# endpoint for Step 1 SQL Injection Task
@bp.route('/login_bypass', methods = ['POST'])
def login_bypass():
    database = get_db()

    user_id = request.get_json()['user_id']
    password = request.get_json()['password']
    # wrong way to compose SQL query based on secure coding practices
    # this allows for SQL Injection to occur
    quote = ""
    if user_id.find("\"") == -1 and user_id.find('\'') == -1:
        user_id = "\"" + user_id + "\""
        password = "\"" + password + "\""
    elif user_id.find('\'') != -1:
        quote = "\'"
    else:
        quote = "\""

    login_q = 'SELECT * FROM USERS WHERE User_ID = {quote}{u_id} AND Password =
       {pwd}'.format(
                            quote=quote, u_id = user_id, pwd = password)

    query_result = database.execute(login_q).fetchall()
    response = {}
    if len(query_result) == NUM_RECORDS_USERS_TABLE:
        response = {
            'isQuerySuccessful': 'true',
            'status': 'SUCCESS',
            'message': 'Congratulations! You successfully used SQL Injection to
              bypass authentication.'
        }

    else:
        response = {
```

```
47              'isQuerySuccessful': 'false',
48              'status': 'ERROR',
49              'message': 'SQL Injection was not successful, please try again.'
50          }
51      print(response)
52      print(user_id)
53      return jsonify(response)
54
55  # endpoint for all SQL Injection after step 1
56  @bp.route('/login_query', methods = ['POST'])
57  def login_query():
58      database = get_db()
59
60      user_id = request.get_json()['user_id']
61      password = request.get_json()['password']
62      game_step = request.get_json()['game_step']
63
64      quote = ""
65      if user_id.find("\"") == -1 and user_id.find('\'') == -1:
66          user_id = "\"" + user_id + "\""
67          password = "\"" + password + "\""
68      elif user_id.find('\'') != -1:
69          quote = "\'"
70      else:
71          quote = "\""
72
73      login_q = 'SELECT * FROM USERS WHERE User_ID = {quote}{u_id} AND Password =
            {quote}{pwd}'.format(
74                              quote=quote, u_id = user_id, pwd = password)
75
76      commands = login_q.split(";", -1)
77      all_query_results = []
78      formatted_query_results = []
79      error = ''
80      try:
81          for command in commands:
82              query_results = database.execute(command).fetchall()
83              all_query_results.append(query_results)
84      except Exception as e:
85          error = e.args
86
87      if error == '':
88          formatted_query_results =''
89          try:
90              table_columns = queried_table_columns(commands[1])
91              formatted_query_results = format_query_results(all_query_results
                    [1], table_columns, game_step)
92          except Exception as e:
93              print(e)
```

```
94              formatted_query_results = 'ERROR'
95          match_expected_results = check_expected_results(all_query_results[1], game_step)
96      if len(formatted_query_results) > 0:
97          if match_expected_results:
98              print_results_to_file(formatted_query_results, game_step)
99              response = {
100                 'isQuerySuccessful': 'true',
101                 'correctResults': 'true',
102                 'results': json.dumps(formatted_query_results),
103                 'error': ''
104             }
105         else:
106             if len(table_columns) > len(CORRECT_RESULTS[game_step][0]):
107                 error = 'SQL Query returns too much information. Follow the directions and be more specific!'
108             else:
109                 error = 'SQL Query was valid but it doesn\'t return the information that you need!'
110             response = {
111                 'isQuerySuccessful': 'true',
112                 'correctResults': 'false',
113                 'results': json.dumps(formatted_query_results),
114                 'error': error
115             }
116     else:
117         error = error if error != '' else 'SQL Query was valid but there were no matching records returned.'
118         response = {
119             'isQuerySuccessful': 'false',
120             'correctResults': 'false',
121             'results': '',
122             'error': error
123         }
124     print(json.dumps(formatted_query_results))
125     print(response)
126     return jsonify(response)
127
128 # endpoint to trigger trojan horse process in step 5
129 @bp.route('/trojan_horse', methods = ['POST'])
130 def trojan_horse():
131     first_name = request.get_json()['first_name']
132     last_name = request.get_json()['last_name']
133
134     if first_name == '':
135         response = {
136             'isSuccess': 'false',
137             'message': 'first name must be provided in order to proceed'
138         }
```

```
139     elif last_name == '':
140         response = {
141             'isSuccess': 'false',
142             'message': 'last name must be provided in order to proceed'
143         }
144     else:
145         execute_trojan_horse(first_name, last_name)
146 
147         response = {
148             'isSuccess': 'true',
149             'message': 'Just a moment! Loading...'
150         }
151 
152     jsonify(response)
153     return response
154 
155 # endpoint that processes submission of suspect guesses
156 @bp.route('/suspect', methods = ['POST'])
157 def suspect():
158     name = request.get_json()['name']
159     game_step = request.get_json()['game_step']
160 
161     correct = check_suspect(name, game_step)
162 
163     response = {}
164     if correct:
165         print_results_to_file(name, game_step)
166         response = {
167             'correct': 'true',
168             'message': 'The evidence suggests that this person is a suspect.'
169         }
170     else:
171         response = {
172             'correct': 'false',
173             'message': 'There isn\'t enough evidence for this person to be a
                suspect.'
174         }
175 
176     jsonify(response)
177     return response
178 
179 # endpoint that processes normal login in final step of the game
180 @bp.route('/login', methods = ['POST'])
181 def login():
182     database = get_db()
183 
184     user_id = request.get_json()['user_id']
185     password = request.get_json()['password']
186     response = {}
```

```
187     if user_id == '':
188         response = {
189             'isLoginSuccessful': 'false',
190             'error': 'You must provide a username'
191         }
192     elif password == '':
193         response = {
194             'isLoginSuccessful': 'false',
195             'error': 'You must provide a password'
196         }
197 
198     quote = ""
199     formatted_password = ''
200     if user_id.find("\"") == -1 and user_id.find('\'') == -1:
201         user_id = "\"" + user_id + "\""
202         formatted_password = "\"" + password + "\""
203     elif user_id.find('\'') != -1:
204         quote = "\'"
205     else:
206         quote = "\""
207 
208     login_q = 'SELECT * FROM USERS WHERE User_ID = {quote}{u_id}' .format
          (quote=quote, u_id = user_id)
209     query_result = database.execute(login_q).fetchone()
210     record = tuple(y for y in query_result)
211     print(record)
212     if len(query_result) == 0:
213         response = {
214             'isLoginSuccessful': 'false',
215             'error': 'Invalid username provided'
216         }
217     else:
218         if password == record[1]:
219             response = {
220                 'isLoginSuccessful': 'true',
221                 'error': ''
222             }
223         else:
224             response =  {
225                 'isLoginSuccessful': 'false',
226                 'error': 'Invalid password provided'
227             }
228     print(response)
229     return response
230 
231 # from database_functions.py
232 # lines 6-33 written by Lia Ferguson
233 # lines 49-91 written by Tom Chmura
234 # dictionary that maps table name to the path of the csv data to populate it
```

```
DB_TABLE_DICT = {
    'BUILDING_ACCESS': 'app/data/BUILDING_ACCESS.csv',
    'COMPUTER_ACCESS': 'app/data/COMPUTER_ACCESS.csv',
    'COMPUTER_TERMINALS': 'app/data/COMPUTER_TERMINALS.csv',
    'QUESTIONNAIRE': 'app/data/QUESTIONNAIRE.csv',
    'USER_INFO': 'app/data/USER_INFO.csv',
    'USERS': 'app/data/USERS.csv',
    'PURCHASE_ORDERS': 'app/data/PURCHASE_ORDERS.csv'
}

# read in data for csv files, and format records properly for insertion into DB
# proper format needed = list of tuples with data in order by columns
# ex. for USERS table
#     return = [(1234, password), (2345, pwd123)]
def get_initial_data(database):
    # path of USERS table csv data
    users_data_path = DB_TABLE_DICT['USERS']
    # SQL query to insert records into users table
    users_insert_query = 'INSERT into USERS (User_ID, Password) VALUES (? , ?)'
    # read in csv data
    with open(users_data_path, newline='\n') as csvfile:
        user_data = csv.reader(csvfile, delimiter=',')
        # skip header row
        next(user_data, None)
        # insert records into database
        for record in user_data:
            database.execute(users_insert_query, record)
            database.commit()

    # path of QUESTIONNAIRE table csv data
    questionnaire_data_path = DB_TABLE_DICT['QUESTIONNAIRE']
    # SQL query to insert records into questionnaire table
    questionnaire_insert_query = 'INSERT into QUESTIONNAIRE (User_ID,
      Favorite_food, Favorite_hobby, Favorite_drink, Allergies) VALUES
      (? , ? , ? , ?, ?)'
    # read in csv data
    with open(questionnaire_data_path, newline='\n') as csvfile:
        questionnaire_data = csv.reader(csvfile, delimiter=',')
        # skip header row
        next(questionnaire_data, None)
        # insert records into database
        for record in questionnaire_data:
            database.execute(questionnaire_insert_query, record)
            database.commit()

    # path of USER_INFO table csv data
    userinfo_data_path = DB_TABLE_DICT['USER_INFO']
    # SQL query to insert records into user info table
    userinfo_insert_query = 'INSERT into USER_INFO (User_ID, First_name,
```

```
          Last_name, Superhero_Name) VALUES (? , ? , ? , ?)'
    # read in csv data
    with open(userinfo_data_path, newline='\n') as csvfile:
        userinfo_data = csv.reader(csvfile, delimiter=',')
        # skip header row
        next(userinfo_data, None)
        # insert records into database
        for record in userinfo_data:
            database.execute(userinfo_insert_query, record)
            database.commit()


    # path of PURCHASE_ORDERS table csv data
    purchaseorders_data_path = DB_TABLE_DICT['PURCHASE_ORDERS']
    # SQL query to insert records into purchase orders table
    purchaseorders_insert_query = 'INSERT into PURCHASE_ORDERS (Po_number,
      User_ID, Item, Cost, Time_received) VALUES (? , ? , ? , ? , ?)'
    # read in csv data
    with open(purchaseorders_data_path, newline='\n') as csvfile:
        purchaseorders_data = csv.reader(csvfile, delimiter=',')
        # skip header row
        next(purchaseorders_data, None)
        # insert records into database
        for record in purchaseorders_data:
            database.execute(purchaseorders_insert_query, record)
            database.commit()


    # path of BUILDING_ACCESS table csv data
    buildingaccess_data_path = DB_TABLE_DICT['BUILDING_ACCESS']
    # SQL query to insert records into building access table
    buildingaccess_insert_query = 'INSERT into BUILDING_ACCESS (Building_ID,
      Building_time, User_ID) VALUES (? , ? , ? )'
    # read in csv data
    with open(buildingaccess_data_path, newline='\n') as csvfile:
        buildingaccess_data = csv.reader(csvfile, delimiter=',')
        # skip header row
        next(buildingaccess_data, None)
        # insert records into database
        for record in buildingaccess_data:
            database.execute(buildingaccess_insert_query, record)
            database.commit()

# from schema.sql
#Written by Tom Chmura
#small updates to USER_INFO, QUESTIONNAIRE, BUILDING_ACCESS, and
  PURCHASE_ORDERS tables by Lia Ferguson

-- Table: BUILDING_ACCESS
```

```
327 CREATE TABLE BUILDING_ACCESS(
328 Building_ID INT NOT NULL,
329 Building_time TIME NOT NULL,
330 User_ID INT NOT NULL,
331 FOREIGN KEY(User_ID) references USERS(User_ID));
332 
333 -- Table: QUESTIONNAIRE
334 CREATE TABLE QUESTIONNAIRE(
335 User_ID INT NOT NULL,
336 Favorite_food VARCHAR(30),
337 Favorite_drink VARCHAR(30),
338 Favorite_hobby VARCHAR(30),
339 Allergies VARCHAR(30),
340 PRIMARY KEY(User_ID)
341 FOREIGN KEY(User_ID) references USERS(User_ID));
342 
343 -- Table: USER_INFO
344 CREATE TABLE USER_INFO(
345 User_ID INT NOT NULL,
346 First_name VARCHAR(20) NOT NULL,
347 Last_name VARCHAR(20),
348 Superhero_Name VARCHAR(30),
349 PRIMARY KEY(User_ID)
350 FOREIGN KEY(User_ID) references USERS(User_ID));
351 
352 -- Table: USERS
353 CREATE TABLE USERS(
354 User_ID INT NOT NULL,
355 Password VARCHAR(30) NOT NULL,
356 PRIMARY KEY(User_ID));
357 
358 -- Table: PURCHASE_ORDERS
359 CREATE TABLE PURCHASE_ORDERS (
360 PO_NUMBER     INT          NOT NULL,
361 USER_ID       INT          NOT NULL,
362 ITEM          VARCHAR (30) NOT NULL,
363 COST          DOUBLE       NOT NULL,
364 TIME_RECEIVED TIME,
365 PRIMARY KEY (PO_NUMBER)
366 FOREIGN KEY(User_ID) references USERS(User_ID));
367 
368 # from helper_functions.py
369 """ code written by Lia Ferguson:
370         all code besides the code written by Andrew Fecher
371 """
372 """code written by Andrew Fecher:
373         line 18, lines 106-117
374 """
375 
```

```
376 # Data structures to hold columns of data tab;es
377 USERS_COLUMNS = ['User_ID', 'Password']
378 USER_INFO_COLUMNS = ['User_ID', 'First_name', 'Last_name', 'Superhero_Name']
379 QUESTIONNAIRE_COLUMNS = ['User_ID', 'Favorite_food', 'Favorite_hobby',
      'Favorite_drink', 'Allergies']
380 PURCHASE_ORDERS_COLUMNS = ['Po_number', 'User_ID', 'Item', 'Cost',
      'Time_Received']
381 BUILDING_ACCESS_COLUMNS = ['Building_ID', 'Building_time', 'User_ID']
382 SQLITE_MASTER_COLUMNS = ['type', 'name', 'tabl_name', 'rootpage', 'sql']
383 #Data structure to hold expected SQL Results
384 CORRECT_RESULTS = {
385     'S3_B1': [('BUILDING_ACCESS',), ('QUESTIONNAIRE',), ('USER_INFO',),
        ('USERS',), ('PURCHASE_ORDERS',)],
386     'S4_B1': [(12592, 'Tony', 'Stark'),
387                         (15687, 'Natasha', 'Romanoff'),
388                         (15685, 'Scott', 'Lang'),
389                         (15972, 'Peter', 'Parker'),
390                         (15423, 'Steve', 'Rogers'),
391                         (15976, 'Thanos', ''),
392                         (17896, 'Bruce', 'Banner')],
393     'S4_B2': [('steak', 'stand-up comedy', 'coffee', 'almonds')],
394     'S5_B1': [(15687, 'almonds'), (17896, 'almonds')],
395     'S5_S': ['Natasha Romanoff', 'Bruce Banner'],
396     'S6_B1': [('Building_ID',), ('Building_time',), ('User_ID',)],
397     'S6_B2': [(15687, '12:55 pm'), (17896, '12:40 pm')],
398     'S6_B3': [(15972, '10:30 am'), (15976, '11:00 am')],
399     'S6_S': ['Peter Parker', 'Thanos'],
400     'S7_B1': [(156834, 15972, 'Coffee Creamer', 5.12, '5:00 pm'),
401                         (156853, 15976, 'Almond Coffee Creamer', 5.23, '5:00
                        pm'),
402                         (438657, 15972, 'Popcorn', 10.25,'12:00pm')],
403     'S7_S': ['Thanos'],
404     'S7_B2': [(15976, 'IAmInevitable')]
405 }
406
407 # Parses out columns that are involved in the SQL Injection Query passed in by
      player
408 def queried_table_columns(query):
409     columns = []
410     columns_and_indices = {} # list that keeps track of column ordering in the
        query
411     if query.casefold().find('questionnaire') != -1:
412         if query.find('*') != -1:
413             columns = QUESTIONNAIRE_COLUMNS
414         else:
415             for column in QUESTIONNAIRE_COLUMNS:
416                 query = query.casefold().partition('from')[0]
417                 if query.find(column.casefold()) != -1:
418                     columns_and_indices[query.find(column.casefold())] = column
```

```
            indices = list(columns_and_indices.keys())
            indices.sort()
            for index in indices:
                columns.append(columns_and_indices[index])
    elif query.casefold().find('user_info') != -1:
        if query.find('*') != -1:
            columns = USER_INFO_COLUMNS
        else:
            for column in USER_INFO_COLUMNS:
                query = query.casefold().partition('from')[0]
                if query.find(column.casefold()) != -1:
                    columns_and_indices[query.find(column.casefold())] = column
            indices = list(columns_and_indices.keys())
            indices.sort()
            for index in indices:
                columns.append(columns_and_indices[index])
    elif query.casefold().find('users') != -1:
        if query.find('*') != -1:
            columns = USERS_COLUMNS
        else:
            for column in USERS_COLUMNS:
                query = query.casefold().partition('from')[0]
                if query.find(column.casefold()) != -1:
                    columns_and_indices[query.find(column.casefold())] = column
            indices = list(columns_and_indices.keys())
            indices.sort()
            for index in indices:
                columns.append(columns_and_indices[index])
    elif query.casefold().find('purchase_orders') != -1:
        if query.find('*') != -1:
            columns = PURCHASE_ORDERS_COLUMNS
        else:
            for column in PURCHASE_ORDERS_COLUMNS:
                query = query.casefold().partition('from')[0]
                if query.find(column.casefold()) != -1:
                    columns_and_indices[query.find(column.casefold())] = column
            indices = list(columns_and_indices.keys())
            indices.sort()
            for index in indices:
                columns.append(columns_and_indices[index])
    elif query.casefold().find('building_access') != -1:
        if query.find('*') != -1:
            columns = BUILDING_ACCESS_COLUMNS
        else:
            for column in BUILDING_ACCESS_COLUMNS:
                query = query.casefold().partition('from')[0]
                if query.find(column.casefold()) != -1:
                    columns_and_indices[query.find(column.casefold())] = column
            indices = list(columns_and_indices.keys())
```

```
468             indices.sort()
469             for index in indices:
470                 columns.append(columns_and_indices[index])
471     elif query.casefold().find('sqlite_master') != -1:
472         if query.find('*') != -1:
473             columns = SQLITE_MASTER_COLUMNS
474         else:
475             for column in SQLITE_MASTER_COLUMNS:
476                 query = query.casefold().partition('from')[0]
477                 if query.find(column.casefold()) != -1:
478                     columns_and_indices[query.find(column.casefold())] = column
479             indices = list(columns_and_indices.keys())
480             indices.sort()
481             for index in indices:
482                 columns.append(columns_and_indices[index])
483 
484     return columns
485 
486 # Formats query results nicely from sqlite3 data structures into dictionaries
487 # for later JSON parsing
488 def format_query_results(query_results, table_columns, game_step):
489     formatted_results = []
490     records = [tuple(y for y in row) for row in query_results]
491     print(records)
492     if len(table_columns) == 0:
493         for record in records:
494             format_record = {}
495             if game_step == 'S6_B1':
496                 format_record['Column'] = record[0]
497             formatted_results.append(format_record)
498     else:
499         for record in records:
500             format_record = {}
501             i = 0
502             for item in record:
503                 format_record[table_columns[i]] = item
504                 i += 1
505             formatted_results.append(format_record)
506     return formatted_results
507 
508 # check whether or not returned records from
509 # SQL query match the expected output
510 def check_expected_results(query_results, game_step):
511     matches_correct_results = False
512     correct_results = CORRECT_RESULTS[game_step]
513     correct_results_length = len(correct_results)
514     comparison = [] # index corresponds to record number, 1 = same, 0 =
        different
515     records = [tuple(y for y in row) for row in query_results]
```

```
516     if len(records) == correct_results_length:
517         for i in range(0, len(records)):
518             sum_match = 0
519             for j in range(0, len(records[0])):
520                 if correct_results[i][j] in records[i]:
521                     sum_match += 1
522             if sum_match == len(records[i]):
523                 comparison.append(1)
524             else:
525                 comparison.append(0)
526         print(comparison)
527         sum_comparison = 0
528         for num in comparison:
529             sum_comparison += num
530         print(sum_comparison)
531         if sum_comparison == len(correct_results):
532             matches_correct_results = True
533     return matches_correct_results
534 
535 # Print results of SQL Injection to Clues.txt file to
536 # assist player with game play
537 def print_results_to_file(formatted_results, game_step):
538     path = os.path.expanduser("~")
539     rest_of_path = ''
540         if platform.system() == 'Windows' :
541         rest_of_path = '\Desktop\SQL-Mystery-Game-Files'
542         clues='\Clues.txt'
543     else:
544         rest_of_path = '/Desktop/Sql-Mystery-Game-Files/'
545         clues='Clues.txt'
546     path += rest_of_path
547     if not os.path.isdir(path):
548         os.mkdir(path)
549     f = open(path + clues, 'a')
550     if game_step == 'S4_B1':
551         f.write("STEP 4 CLUES\n")
552         f.write("Employee User ID\'s\n\n")
553     elif game_step == 'S4_B2':
554         f.write("Tony Stark's Questionnaire Data\n\n")
555     elif game_step == 'S5_B1':
556         f.write("STEP 5 CLUES\n")
557         f.write("Discover Possible Almond Snackers\n\n")
558     elif game_step == 'S5_S':
559         f.write("Suspect\n\n")
560     f.write(json.dumps(formatted_results, indent=4, sort_keys=False))
561     f.write('\n\n')
562     f.close()
563 
564 # execute "trojan horse" - download
```

```
565 # confession file onto player's computer with their name filled in
566 def execute_trojan_horse(first_name, last_name):
567     path = os.path.expanduser("~")
568     rest_of_path = ''
569     if(platform.system() == 'Windows'):
570         rest_of_path = '\Desktop\SQL-Mystery-Game-Files\Confidential'
571         file_path = '\For Police.txt'
572     else:
573         rest_of_path = '/Desktop/Sql-Mystery-Game-Files/Confidential'
574         file_path = '/For Police.txt'
575     path += rest_of_path
576 
577     os.mkdir(path)
578     f_read = open('app/data/trojan_horse_confess_template.txt', 'r')
579     f_write = open(path + file_path, 'a')
580     f_write.write(f_read.read())
581     f_write.write(first_name + " " + last_name)
582 
583     f_read.close()
584     f_write.close()
585 
586 # verify if the suspect entered by player is correct
587 def check_suspect(name, game_step):
588     correct = False
589     suspects = CORRECT_RESULTS[game_step]
590     for suspect in suspects:
591         if name.casefold() == suspect.casefold():
592             correct = True
593             break
594     return correct
595 
```