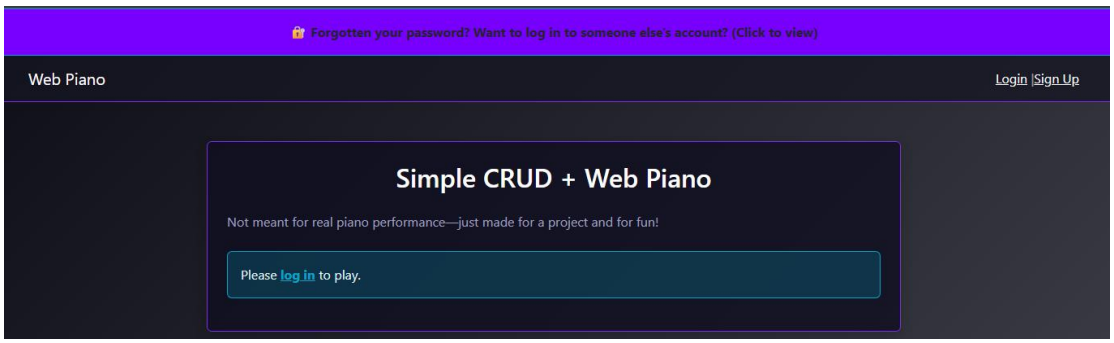
Presentation Outline: Web Piano with CRUD Operations

Link for the project: *https://takonyishi.infinityfreeapp.com*

Overview



Purpose:

A web based piano application with user authentication (CRUD) And interactive gameplay.

Features:

User registration/Sign up(Create)
Login (Read)
Account management (Update/Delete)
Play mode (Piano challenges)
Free Play mode (Piano gameplay or idk freeplay itself)

Tech Stack:

Frontend: React (via Babel), Bootstrap CSS, Tone.js(audio)
Backend: PHP, MySQL

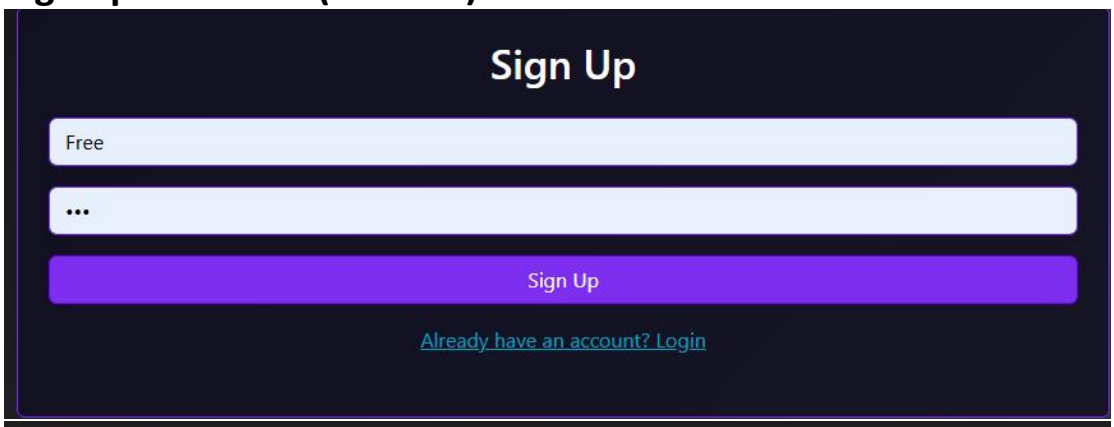
Core Functions

Database Connection + Free hosting setup

```php
<?php
session_start();

$mode = $_GET['mode'] ?? 'menu';
$title = match($mode) {
  'play'   => "Play Mode",
  'free'   => "Free Play",
  'login'  => "Login",
  'signup' => "Sign Up",
  default  => "Simple CRUD + Web Piano"
};

// Database credentials
$db_host = "sql302.infinityfree.com";
$db_user = "if0_38869516";
$db_pass = "BryNyMic";
$db_name = "if0_38869516_dbplayers";

$conn = new mysqli(hostname: $db_host, username: $db_user, password: $db_pass, database: $db_name);
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);
```

CRUD Operations

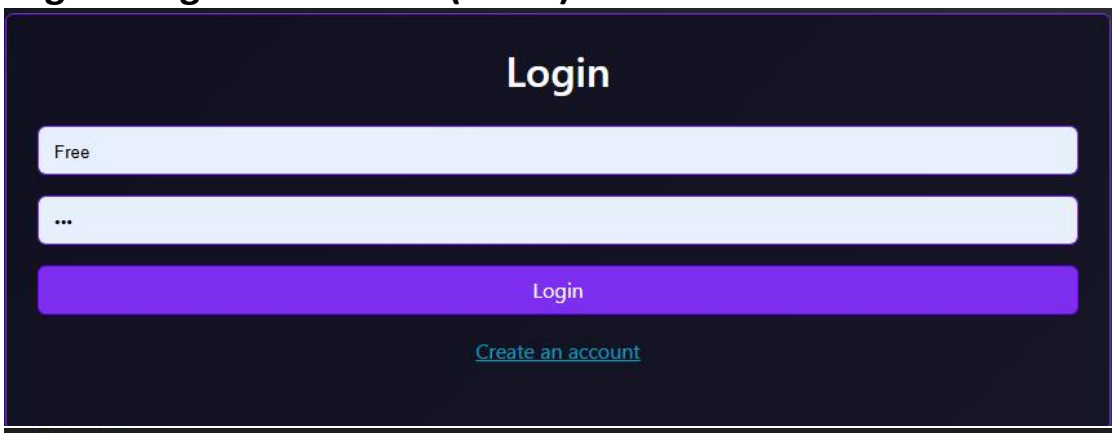
Sign up structure:(CREATE)




```php
// Handle signup
$signup_error = "";
if ($_SERVER["REQUEST_METHOD"] === "POST" && $mode === "signup") {
  $username = $_POST["username"];
  $raw_password = $_POST["password"];
  $password = password_hash(password: $raw_password, algo: PASSWORD_DEFAULT);

  // Ensure username is unique
  $stmt = $conn->prepare(query: "SELECT username FROM players WHERE username=? LIMIT 1");
  $stmt->bind_param(types: "s", var: &$username);
  $stmt->execute();
  $result = $stmt->get_result();

  if ($result->num_rows === 0) {
    $stmt = $conn->prepare(query: "INSERT INTO players (username, password, raw_password) VALUES (?, ?, ?)");
    $stmt->bind_param(types: "sss", var: &$username, vars: &$password, $raw_password);

    if ($stmt->execute()) {
      header(header: "Location: index.php?mode=login");
      exit;
    } else {
      $signup_error = "Error creating account. Please try again.";
    }
  } else {
    $signup_error = "Username already taken. Please choose another.";
  }
}
```

```jsx
// SignupForm component
function SignupForm({ error, mode }) {
  return (
    <form method="post" className="mb-4">
      <div className="mb-3">
        <input type="text" name="username" className="form-control" placeholder="Username" required />
      </div>
      <div className="mb-3">
        <input type="password" name="password" className="form-control" placeholder="Password" required />
      </div>
      <button type="submit" className="btn btn-primary w-100">Sign Up</button>
      {error && <div className="alert alert-danger mt-3">{error}</div>}
      <div className="text-center mt-3">
        <a href="?mode=login">Already have an account? Login</a>
      </div>
    </form>
  );
}
```

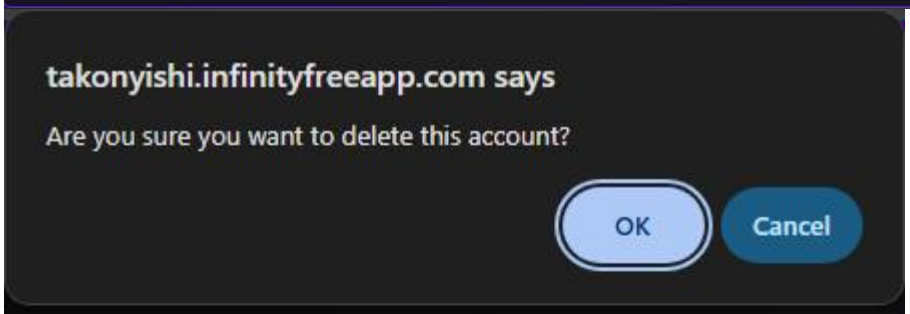
Login + Logout Structure:(READ)

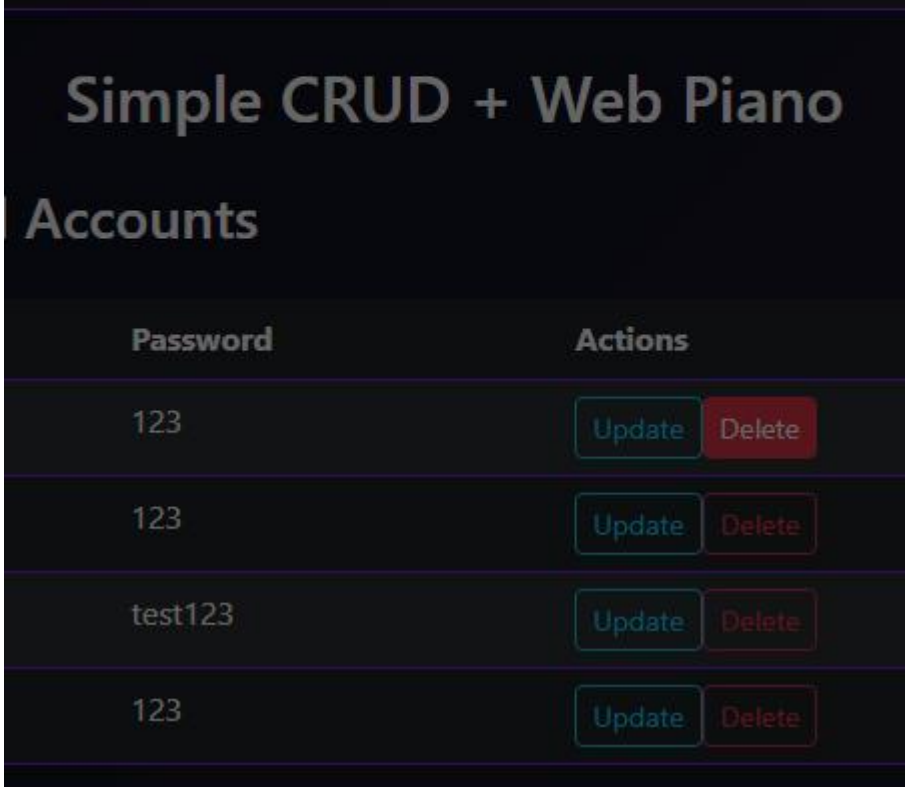


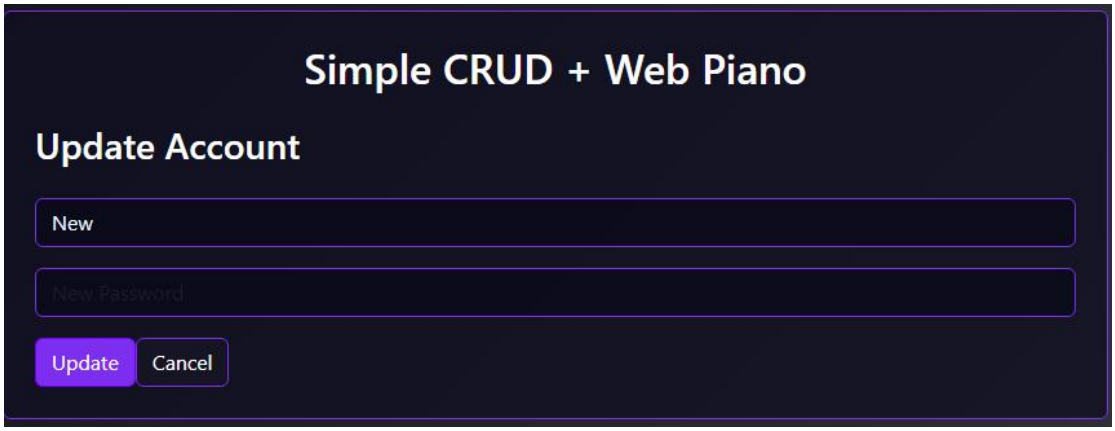

```php
// Handle login
$login_error = "";
if ($_SERVER["REQUEST_METHOD"] === "POST" && $mode === "login") {
  $username = $_POST["username"];
  $password = $_POST["password"];
  $stmt = $conn->prepare(query: "SELECT password FROM players WHERE BINARY username=? LIMIT 1");
  $stmt->bind_param(types: "s", var: &$username);
  $stmt->execute();
  $result = $stmt->get_result();

  if ($result->num_rows === 1) {
    $row = $result->fetch_assoc();
    if (password_verify(password: $password, hash: $row["password"])) {
      $_SESSION['username'] = $username;
      header(header: "Location: index.php?mode=menu");
      exit;
    } else {
      $login_error = "Invalid login. Please try again.";
    }
  } else {
    $login_error = "Invalid login. Please try again.";
  }
}

// Handle logout
if (isset($_GET['logout'])) {
  session_destroy();
  header(header: "Location: index.php?mode=menu");
  exit;
}
```

```jsx
// LoginForm component
function LoginForm({ error, mode }) {
  return (
    <form method="post" className="mb-4">
      <div className="mb-3">
        <input type="text" name="username" className="form-control" placeholder="Username" required />
      </div>
      <div className="mb-3">
        <input type="password" name="password" className="form-control" placeholder="Password" required />
      </div>
      <button type="submit" className="btn btn-primary w-100">Login</button>
      {error && <div className="alert alert-danger mt-3">{error}</div>}
      <div className="text-center mt-3">
        <a href="?mode=signup">Create an account</a>
      </div>
    </form>
  );
}
```

UPDATE + DELETE

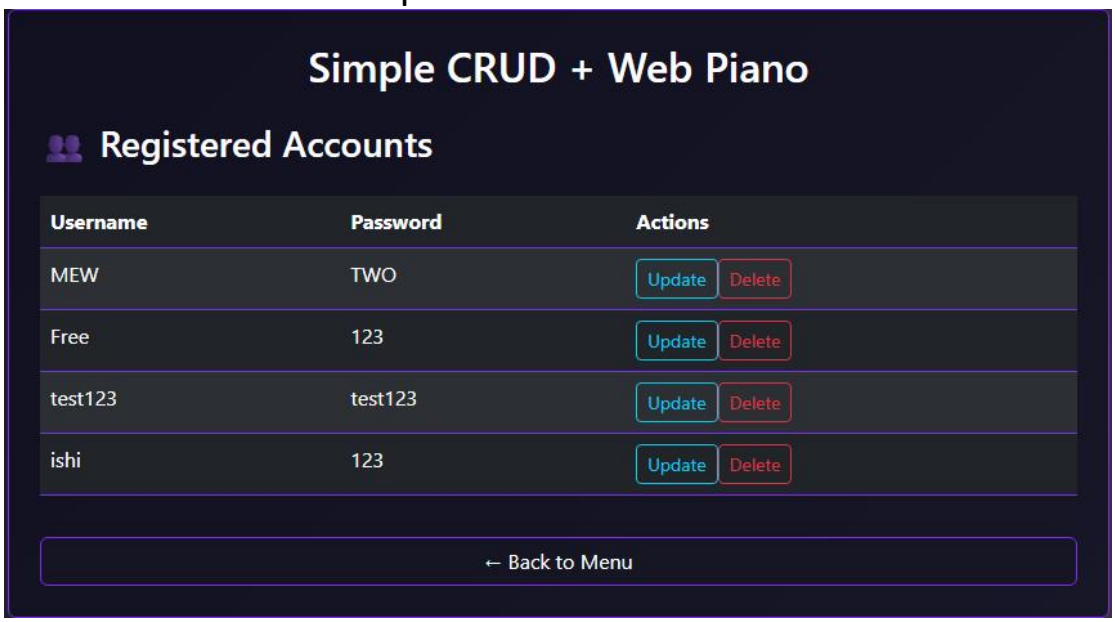






Can change either Password or Username

Was New 123 before updated to New Mew



Was New mew before updated to MEW TWO

As we can see on the new Registered Accounts the gmail username got removed(deleted) and the New 123 got updated

Update and Delete code Structure:

```php
// Handle Update form display
if ($mode === 'update' && isset($_GET['id'])) {
  $update_id = (int)$_GET['id'];
  $result = $conn->query(query: "SELECT id, username, raw_password FROM players WHERE id=$update_id");
  $update_user = $result->fetch_assoc();
}

// Handle Update form submit
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST["update_user"])) {
  $update_id = (int)$_POST["id"];
  $new_username = $_POST["username"];
  $new_password = $_POST["password"];
  $hashed_password = password_hash(password: $new_password, algo: PASSWORD_DEFAULT);

  $stmt = $conn->prepare(query: "UPDATE players SET username=?, password=?, raw_password=? WHERE id=?");
  $stmt->bind_param(types: "sssi", var: &$new_username, vars: &$hashed_password, $new_password, $update_id);
  $stmt->execute();

  header(header: "Location: index.php?mode=accounts");
  exit;
}
```

```php
// Handle Delete
if ($mode === 'delete' && isset($_GET['id'])) {
    $id = (int)$_GET['id'];
    $stmt = $conn->prepare(query: "DELETE FROM players WHERE id=?");
    $stmt->bind_param(types: "i", var: &$id);
    $stmt->execute();
    header(header: "Location: index.php?mode=accounts");
    exit;
  }
```

```php
if ($mode === 'accounts' || $mode === 'update') {
  $result = $conn->query(query: "SELECT id, username, raw_password FROM players ORDER BY created_at DESC");
  $react_data['accounts'] = $result->fetch_all(mode: MYSQLI_ASSOC);
}
?>
```

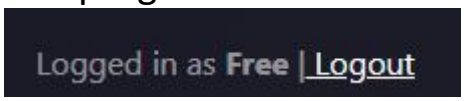
```
// Accounts component
function Accounts({ accounts, updateUser }) {
  if (appData.mode === 'update' && updateUser) {
    return (
      <>
        <h3 className="mb-4">Update Account</h3>
        <form method="post">
          <input type="hidden" name="id" value={updateUser.id} />
          <div className="mb-3">
            <input
              type="text"
              name="username"
              className="form-control"
              defaultValue={updateUser.username}
              required
            />
          </div>
          <div className="mb-3">
            <input
              type="password"
              name="password"
              className="form-control"
              placeholder="New Password"
              required
            />
          </div>
          <button type="submit" name="update_user" className="btn btn-primary">Update</button>
          <a href="?mode=accounts" className="btn btn-secondary">Cancel</a>
        </form>
      </>
    );
  } else {
```


```
// Accounts component
function Accounts({ accounts, updateUser }) {
  if (appData.mode === 'update' && updateUser) {
    return (
      <>
        <h3 className="mb-4">Update Account</h3>
        <form method="post">
          <input type="hidden" name="id" value={updateUser.id} />
          <div className="mb-3">
            <input
              type="text"
              name="username"
              className="form-control"
              defaultValue={updateUser.username}
              required
            />
          </div>
          <div className="mb-3">
            <input
              type="password"
              name="password"
              className="form-control"
              placeholder="New Password"
              required
            />
          </div>
          <button type="submit" name="update_user" className="btn btn-primary">Update</button>
          <a href="?mode=accounts" className="btn btn-secondary">Cancel</a>
        </form>
      </>
    );
  } else {
```

```
    return (
      <>
        <h3 className="mb-4">👥 Registered Accounts</h3>
        <div className="table-responsive">
          <table className="table table-dark table-striped">
            <thead>
              <tr>
                <th>Username</th>
                <th>Password</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {accounts.map(account => (
                <tr key={account.id}>
                  <td>{account.username}</td>
                  <td>{account.raw_password}</td>
                  <td>
                    <a href={`?mode=update&id=${account.id}`} className="btn btn-sm btn-outline-info">Update</a>
                    <a
                      href={`?mode=delete&id=${account.id}`}
                      className="btn btn-sm btn-outline-danger"
                      onClick={(e) => {
                        if (!confirm('Are you sure you want to delete this account?')) {
                          e.preventDefault();
                        }
                      }}
                    >
                      Delete
                    </a>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
        <a href="?mode=menu" className="btn btn-secondary mt-3">← Back to Menu</a>
      </>
    );
  }
}
```


```
    return (
      <>
        <h3 className="mb-4">👥 Registered Accounts</h3>
        <div className="table-responsive">
          <table className="table table-dark table-striped">
            <thead>
              <tr>
                <th>Username</th>
                <th>Password</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {accounts.map(account => (
                <tr key={account.id}>
                  <td>{account.username}</td>
                  <td>{account.raw_password}</td>
                  <td>
                    <a href={`?mode=update&id=${account.id}`} className="btn btn-sm btn-outline-info">Update</a>
                    <a
                      href={`?mode=delete&id=${account.id}`}
                      className="btn btn-sm btn-outline-danger"
                      onClick={(e) => {
                        if (!confirm('Are you sure you want to delete this account?')) {
                          e.preventDefault();
                        }
                      }}
                    >
                      Delete
                    </a>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
        <a href="?mode=menu" className="btn btn-secondary mt-3">← Back to Menu</a>
      </>
    );
  }
}
```

The program also tracks on which account ur logged onto/into

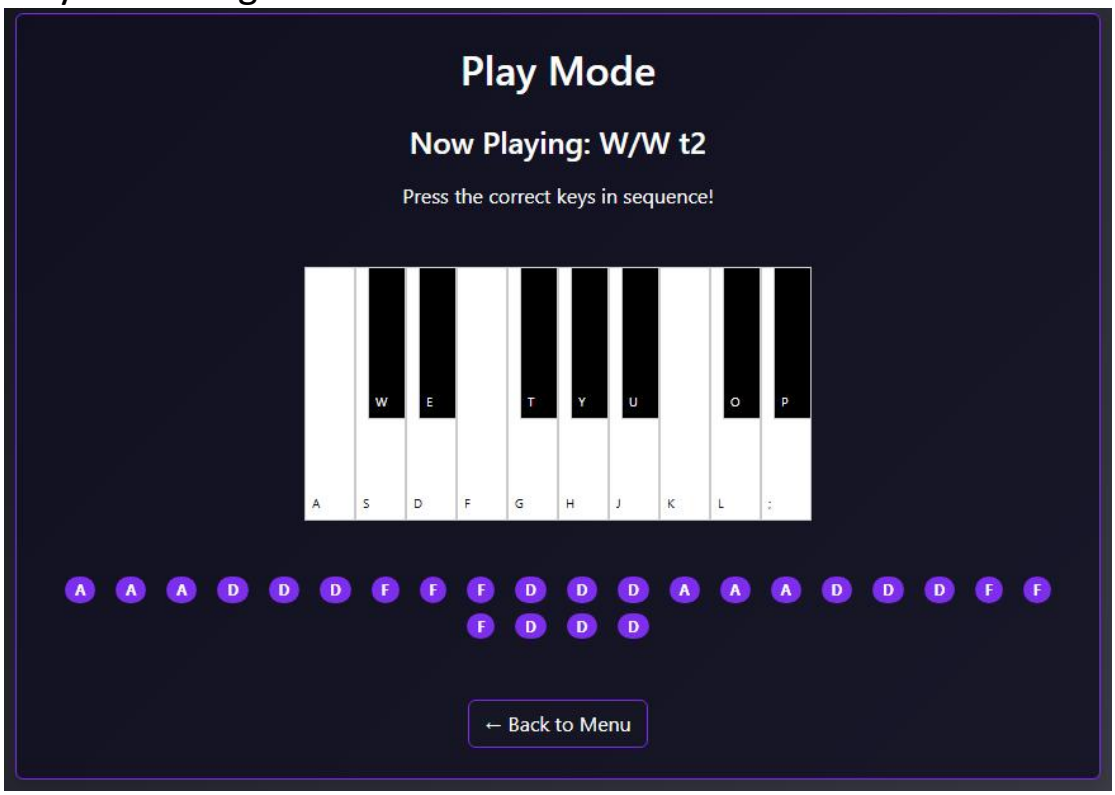


Counts as READ ig

How the piano part of the code works

Game modes:

Play Mode Logic

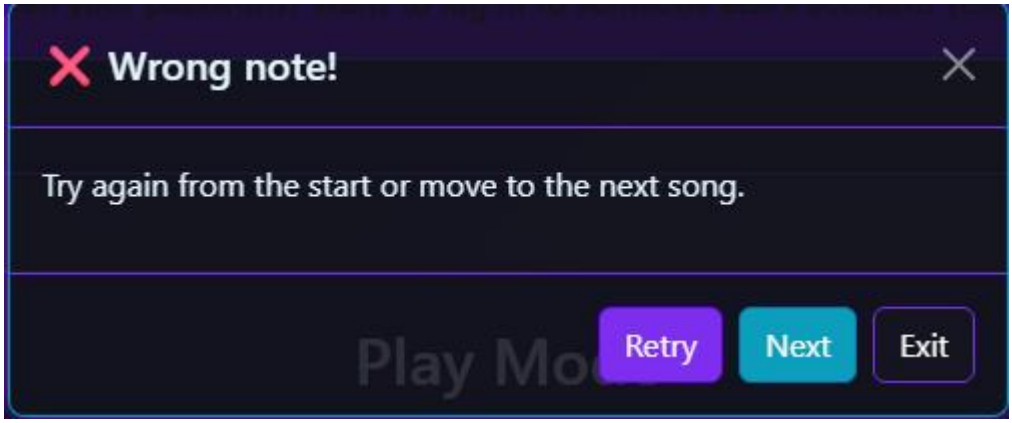


Randomly selects a song from songs array

```
const songs = [
  { title: "Twinkle Twinkle", notes: "a a g g h h g f f d d s s a g g f f d d s g g f f d d s a a g g h h g f f d d s s a" },
  { title: "Happy Birthday? ig", notes: "a a k a ; l a a k a p ; a a p ; l ; k" },
  { title: "Mary Had a Little Lamb", notes: "d s a s d d d s s s d f f d s a s d d d d s s d s a" },
  { title: "W/W t1", notes: "s s d f f d s a s d d a a s s d f f d s a s d a s" },
  { title: "Jingle Bells", notes: "s s s s s s s d a s f g s s s s s s s d a s g f" },
  { title: "W/W t2", notes: "a a a d d d f f f d d d a a a d d d f f f d d d" },
  { title: "W/B t1", notes: "e d e d e j d k h" },
  { title: "W/B t2", notes: "a w a w s e s e d t d t f y f y g u g u" },
  { title: "W/B t3", notes: "d d o d l k j o p l" },
  { title: "W/B t4", notes: "a a d d f f g e t y u" },
  { title: "W/B t5", notes: "d d d g d a d d g d a d f d d f d d d f d d f" },
  { title: "W/B t6", notes: "a w a w a w s e s e s e d t d t d t f y f y f y" }
];
```

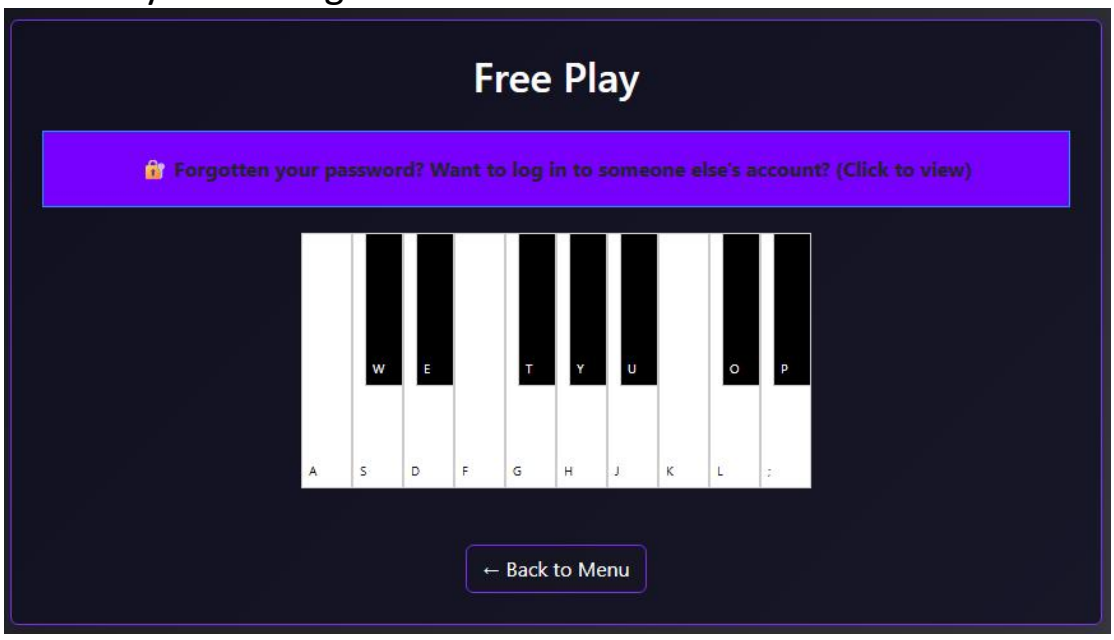
Tracks the user input against the songs note sequence

Shows the progress with colored badges(blue)



If user/player enters/presses anything else besides D or the first note for that specific song it'll counts as fail thus user/player can choose between retry(back to zero progress) next(next song and yes it is random from the list not twinkle twinkle to next) exit(it works how it sounds)

Free Play Mode Logic



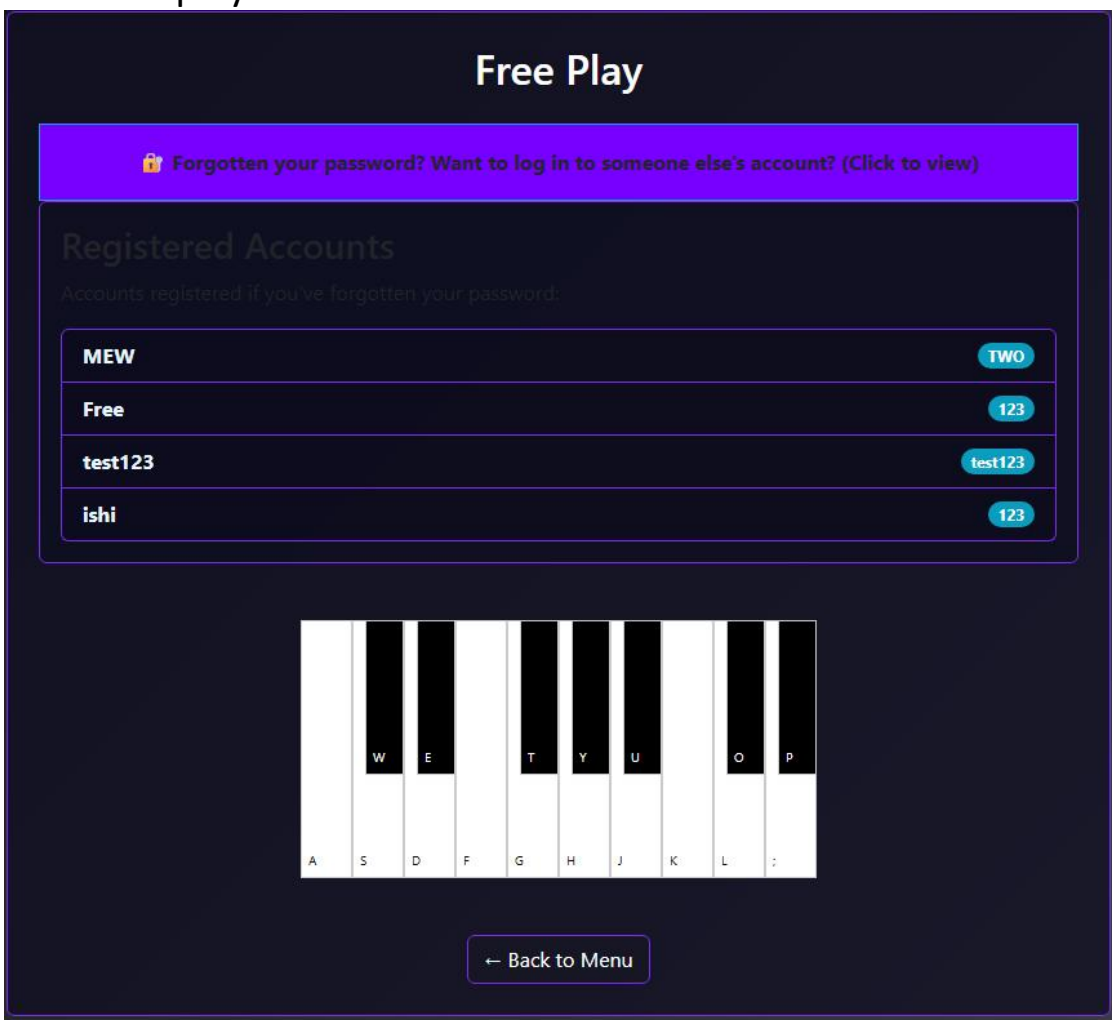
No rules user/player can play whatever he/she wants

Displays registered accounts (since it is for demo + project purposes I don’t think this will harm other users as this project has no leaderboards or whatever)

Tone.js handles the audio (not meant to be accurate as again this is for demo + project purposes)

Q: Why does the click to view in the same page as ur free play mode?

A: For less web pages as currently we have the Main page, See accounts(Update/Delete) page, Login page, Signup page, Play Mode page and thus we decided to make the click to view into same page as Free Play for logged in and out users/players to access free play and as stated there “Forgotten your password? Want to log in to someone else’s account?” for em to access both free play mode and the click to view mode



Code Structure:

```jsx
// Menu component
function Menu({ username }) {
  return (
    <>
      <p className="text-muted mb-4">Not meant for real piano performance—just made for a project and for fun!</p>
      {!username ? (
        <div className="alert alert-info">
          Please <a href="?mode=login" className="alert-link">log in</a> to play.
        </div>
      ) : (
        <div className="d-grid gap-3">
          <button
            onClick={() => window.location.href='?mode=play'}
            className="btn btn-primary btn-lg py-3"
          >
            Play Mode
          </button>
          <button
            onClick={() => window.location.href='?mode=free'}
            className="btn btn-success btn-lg py-3"
          >
            Free Play
          </button>
          <button
            onClick={() => window.location.href='?mode=accounts'}
            className="btn btn-info btn-lg py-3"
          >
            See Accounts
          </button>
        </div>
      )}
    </>
  );
}
```

```jsx
// FreePlay component
function FreePlay({ freeAccounts }) {
  const [activeKeys, setActiveKeys] = React.useState(new Set());
  const [showAccounts, setShowAccounts] = React.useState(false);

  const handleNotePlay = (note) => {
    synth.triggerAttack(note);
  };

  return (
    <>
      <div className="alert alert-warning text-center mb-0 rounded-0">
        <button
          className="text-decoration-none text-dark fw-bold border-0 bg-transparent w-100"
          onClick={() => setShowAccounts(!showAccounts)}
        >
          🔐 Forgotten your password? Want to log in to someone else's account? (Click to view)
        </button>
      </div>

      {showAccounts && (
        <div id="free-accounts" className="card bg-secondary mb-4">
          <div className="card-body">
            <h3 className="card-title">Registered Accounts</h3>
            <p className="card-text">Accounts registered if you've forgotten your password:</p>
            <ul className="list-group">
              {freeAccounts.map((account, index) => (
                <li key={index} className="list-group-item d-flex justify-content-between align-items-center">
                  <strong>{account.username}</strong>
                  <span className="badge bg-primary rounded-pill">{account.raw_password}</span>
                </li>
              ))}
            </ul>
          </div>
        </div>
      )}
```

```
        <Piano mode="free" onNotePlay={handleNotePlay} />

        <div className="text-center mt-4">
          <a href="?mode=menu" className="btn btn-secondary">← Back to Menu</a>
        </div>
      </>
    );
  }
```

```
// Main App component
function App() {
  const { mode, title, username, signup_error, login_error, free_accounts, accounts, update_user } = appData

  return (
    <div className="d-flex flex-column min-vh-100">
      {/* Alert bar at top */}
      <div className="alert alert-warning text-center mb-0 rounded-0">
        <a href="?mode=free#free-accounts" className="text-decoration-none text-dark fw-bold">
          🔐 Forgotten your password? Want to log in to someone else's account? (Click to view)
        </a>
      </div>

      {/* Navigation bar */}
      <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
        <div className="container">
          <a className="navbar-brand" href="?mode=menu">Web Piano</a>
          <div className="navbar-text ms-auto">
            {username ? (
              <>
                Logged in as <strong>{username}</strong> |
                <a href="?logout=1" className="text-white"> Logout</a>
              </>
            ) : (
              <>
                <a href="?mode=login" className="text-white">Login</a> |
                <a href="?mode=signup" className="text-white">Sign Up</a>
              </>
            )}
          </div>
        </div>
      </nav>
```

```
      <div className="container my-5 flex-grow-1">
        <div className="row justify-content-center">
          <div className="col-lg-8">
            <div className="card bg-dark text-white p-4 shadow">
              <h2 className="card-title text-center mb-4">{title}</h2>

              {mode === 'signup' ? (
                <SignupForm error={signup_error} mode={mode} />
              ) : mode === 'login' ? (
                <LoginForm error={login_error} mode={mode} />
              ) : mode === 'accounts' || mode === 'update' ? (
                username ? (
                  <Accounts accounts={accounts} updateUser={update_user} />
                ) : (
                  <div className="alert alert-warning">
                    You must be logged in to view accounts. <a href="?mode=login" className="alert-link">Go to Login</a>
                  </div>
                )
              ) : mode === 'menu' ? (
                <Menu username={username} />
              ) : mode === 'play' ? (
                username ? (
                  <PlayMode />
                ) : (
                  <div className="alert alert-warning">
                    You must be logged in. <a href="?mode=login" className="alert-link">Go to Login</a>
                  </div>
                )
              ) : mode === 'free' ? (
                <FreePlay freeAccounts={free_accounts} />
              ) : null}
            </div>
          </div>
        </div>
      </div>
    );
  }
```

```
// Handle Free Accounts + free play display
if ($mode === 'free') {
  $result = $conn->query(query: "SELECT username, raw_password FROM players ORDER BY created_at DESC LIMIT 5");
  $free_accounts = $result->fetch_all(mode: MYSQLI_ASSOC);
} else {
  $free_accounts = [];
}
```

```
// PlayMode component
function PlayMode() {
  const [currentSongIndex, setCurrentSongIndex] = React.useState(-1);
  const [completedSongs, setCompletedSongs] = React.useState(new Set());
  const [currentStep, setCurrentStep] = React.useState(0);
  const [songNotes, setSongNotes] = React.useState([]);
  const [showCompletionModal, setShowCompletionModal] = React.useState(false);
  const [showErrorModal, setShowErrorModal] = React.useState(false);

  const pickRandomSong = React.useCallback(() => {
    const remaining = songs.map((_, i) => i).filter(i => !completedSongs.has(i));
    if (remaining.length === 0) {
      setCompletedSongs(new Set());
      return pickRandomSong();
    }
    return remaining[Math.floor(Math.random() * remaining.length)];
  }, [completedSongs]);

  const loadSong = React.useCallback((idx) => {
    setCurrentSongIndex(idx);
    setSongNotes(songs[idx].notes.split(' '));
    setCurrentStep(0);
  }, []);

  React.useEffect(() => {
    loadSong(pickRandomSong());
  }, [loadSong, pickRandomSong]);

  const handleNotePlay = React.useCallback((note, key) => {
    synth.triggerAttack(note);

    const expected = songNotes[currentStep];
    const playedKey = key || notes.find(n => n.note === note)?.key;

    if (playedKey === expected) {
      const newStep = currentStep + 1;
      setCurrentStep(newStep);

      if (newStep === songNotes.length) {
        setShowCompletionModal(true);
      }
    } else {
      setShowErrorModal(true);
    }
  }, [currentStep, songNotes]);
```

```
  const updateProgress = React.useCallback(() => {
    return songNotes.map((k, i) => (
      <span
        key={`${i}-${k}`}
        className={`badge rounded-pill mx-1 ${i < currentStep ? 'bg-success' : 'bg-secondary'}`}
      >
        {k.toUpperCase()}
      </span>
    ));
  }, [songNotes, currentStep]);

  const handleReplay = () => {
    loadSong(currentSongIndex);
    setShowCompletionModal(false);
  };

  const handleNext = () => {
    const newCompleted = new Set(completedSongs);
    newCompleted.add(currentSongIndex);
    setCompletedSongs(newCompleted);
    loadSong(pickRandomSong());
    setShowCompletionModal(false);
  };

  const handleRetry = () => {
    loadSong(currentSongIndex);
    setShowErrorModal(false);
  };

  const handleNextAfterError = () => {
    loadSong(pickRandomSong());
    setShowErrorModal(false);
  };
```

```
  return (
    <>
      <h4 className="text-center mb-3">Now Playing: {songs[currentSongIndex]?.title || 'Loading...'}</h4>
      <p className="text-center mb-4">Press the correct keys in sequence!</p>

      <Piano mode="play" onNotePlay={handleNotePlay} />

      <div className="d-flex flex-wrap justify-content-center gap-2 my-4">
        {updateProgress()}
      </div>

      <div className="text-center mt-4">
        <a href="?mode=menu" className="btn btn-secondary">← Back to Menu</a>
      </div>

      {/* Completion Modal */}
      <div className={`modal fade ${showCompletionModal ? 'show' : ''}`} style={{ display: showCompletionModal ? 'block' : 'none' }}>
        <div className="modal-dialog">
          <div className="modal-content">
            <div className="modal-header">
              <h5 className="modal-title">🎉 You completed the song!</h5>
              <button type="button" className="btn-close" onClick={() => setShowCompletionModal(false)}></button>
            </div>
            <div className="modal-body">
              <p>Great job! Would you like to try again or move to the next song?</p>
            </div>
            <div className="modal-footer">
              <button className="btn btn-primary" onClick={handleReplay}>Retry</button>
              <button className="btn btn-success" onClick={handleNext}>Next</button>
              <a href="?mode=menu" className="btn btn-secondary">Exit</a>
            </div>
          </div>
        </div>
      </div>
      {showCompletionModal && <div className="modal-backdrop fade show"></div>}
```

```jsx
      {/* Error Modal */}
      <div className={`modal fade ${showErrorModal ? 'show' : ''}`} style={{ display: showErrorModal ? 'block' : 'none' }}>
        <div className="modal-dialog">
          <div className="modal-content">
            <div className="modal-header">
              <h5 className="modal-title">❌ Wrong note!</h5>
              <button type="button" className="btn-close" onClick={() => setShowErrorModal(false)}></button>
            </div>
            <div className="modal-body">
              <p>Try again from the start or move to the next song.</p>
            </div>
            <div className="modal-footer">
              <button className="btn btn-primary" onClick={handleRetry}>Retry</button>
              <button className="btn btn-success" onClick={handleNextAfterError}>Next</button>
              <a href="?mode=menu" className="btn btn-secondary">Exit</a>
            </div>
          </div>
        </div>
      </div>
      {showErrorModal && <div className="modal-backdrop fade show"></div>}
    </>
  );
}
```

Frontend Components

React Integration

Data Flow: PHP passes user sessions errors etcs into React via json_encode()

Key components:

Piano() Renders keyboard and handles playbacks
Accounts() Displays user with Update/Delete function/buttons
LoginForm/SignupForm() Authentication forms

Tone.js Audio will be shown in the Piano components
Use of synth.triggerName

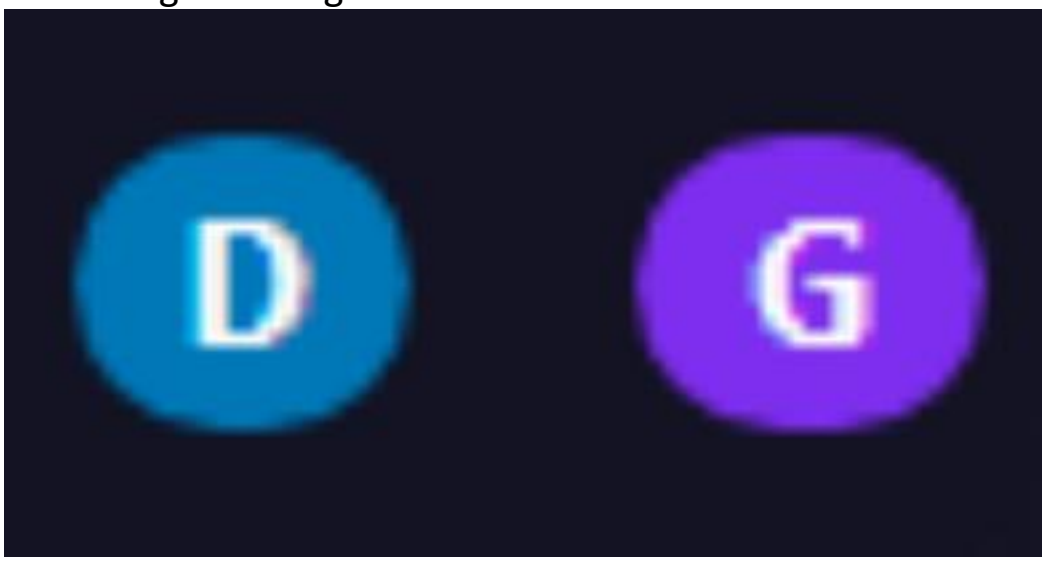
Styling

Theme: Dark mode + Purple(raiden) + Blueish(shorekeeper) accents

Responsive: Bootstrap grid + customized CSS for piano keys

Visual Feedbacks: Active glows (box - shadow)

Progress badges turn blue when correct



Piano Components:

```
    // Piano component
function Piano({ mode, onNotePlay }) {
  const keyMap = React.useMemo(() => {
    const map = {};
    notes.forEach(note => {
      map[note.key] = note.note;
    });
    return map;
  }, []);

  const [activeKeys, setActiveKeys] = React.useState(new Set());

  const getBlackKeyPosition = (note) => {
    const whiteKeysBefore = notes.filter(n => !n.black &&
      notes.indexOf(n) < notes.findIndex(n => n.note === note));
    return whiteKeysBefore.length * 40 + 25;
  };

  const handleKeyDown = React.useCallback((e) => {
    const key = e.key;
    if (!keyMap[key] || activeKeys.has(key)) return;

    setActiveKeys(prev => {
      const newSet = new Set(prev);
      newSet.add(key);
      return newSet;
    });

    const note = keyMap[key];
    if (note) {
      onNotePlay(note, key);
    }
  }, [keyMap, activeKeys, onNotePlay]);

  const handleKeyUp = React.useCallback((e) => {
    const key = e.key;
    if (activeKeys.has(key)) {
      setActiveKeys(prev => {
        const newSet = new Set(prev);
        newSet.delete(key);
        return newSet;
      });
```

```
      const note = keyMap[key];
      if (note) {
        synth.triggerRelease();
      }
    }
  }, [keyMap, activeKeys]);

  React.useEffect(() => {
    window.addEventListener('keydown', handleKeyDown);
    window.addEventListener('keyup', handleKeyUp);
    return () => {
      window.removeEventListener('keydown', handleKeyDown);
      window.removeEventListener('keyup', handleKeyUp);
    };
  }, [handleKeyDown, handleKeyUp]);

  const handleMouseDown = (note) => {
    synth.triggerAttack(note);
    onNotePlay(note);
  };

  const handleMouseUp = (note) => {
    synth.triggerRelease();
  };
```

```jsx
return (
  <div className="piano">
    <div className="keys-container">
      <div className="white-keys">
        {notes.filter(n => !n.black).map(note => (
          <div
            key={note.note}
            className={`key ${activeKeys.has(note.key) ? 'active' : ''}`}
            data-key={note.key}
            data-note={note.note}
            onMouseDown={() => handleMouseDown(note.note)}
            onMouseUp={() => handleMouseUp(note.note)}
            onMouseLeave={() => handleMouseUp(note.note)}
            onTouchStart={(e) => {
              e.preventDefault();
              handleMouseDown(note.note);
            }}
            onTouchEnd={() => handleMouseUp(note.note)}
          >
            <span className="label">{note.key.toUpperCase()}</span>
          </div>
        ))}
      </div>
      <div className="black-keys">
        {notes.filter(n => n.black).map(note => (
          <div
            key={note.note}
            className={`key black ${activeKeys.has(note.key) ? 'active' : ''}`}
            style={{ left: `${getBlackKeyPosition(note.note)}px` }}
            data-key={note.key}
            data-note={note.note}
            onMouseDown={() => handleMouseDown(note.note)}
            onMouseUp={() => handleMouseUp(note.note)}
            onMouseLeave={() => handleMouseUp(note.note)}
            onTouchStart={(e) => {
              e.preventDefault();
              handleMouseDown(note.note);
            }}
            onTouchEnd={() => handleMouseUp(note.note)}
          >
            <span className="label">{note.key.toUpperCase()}</span>
          </div>
        ))}
      </div>
    </div>
  </div>
```

PHP to React snippets:

```php
// Prepare data for React
$react_data = [
    'mode' => $mode,
    'title' => $title,
    'username' => $_SESSION['username'] ?? null,
    'signup_error' => $signup_error,
    'login_error' => $login_error,
    'free_accounts' => $free_accounts,
    'update_user' => $update_user ?? null
];
```

```php
if ($mode === 'accounts' || $mode === 'update') {
    $result = $conn->query(query: "SELECT id, username, raw_password FROM players ORDER BY created_at DESC");
    $react_data['accounts'] = $result->fetch_all(mode: MYSQLI_ASSOC);
}
?>
<!-- React dependencies -->
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

```php
    // Pass PHP data to React
    const appData = <?= json_encode(value: $react_data) ?>;
```

```javascript
    // Piano component
function Piano({ mode, onNotePlay }) {
    const keyMap = React.useMemo(() => {
        const map = {};
        notes.forEach(note => {
            map[note.key] = note.note;
        });
        return map;
    }, []);

const [activeKeys, setActiveKeys] = React.useState(new Set());

const getBlackKeyPosition = (note) => {
    const whiteKeysBefore = notes.filter(n => !n.black &&
        notes.indexOf(n) < notes.findIndex(n => n.note === note));
    return whiteKeysBefore.length * 40 + 25;
};

const handleKeyDown = React.useCallback((e) => {
    const key = e.key;
    if (!keyMap[key] || activeKeys.has(key)) return;
```

```
React.useEffect(() => {
  window.addEventListener('keydown', handleKeyDown);
  window.addEventListener('keyup', handleKeyUp);
  return () => {
    window.removeEventListener('keydown', handleKeyDown);
    window.removeEventListener('keyup', handleKeyUp);
  };
}, [handleKeyDown, handleKeyUp]);
```

```
// PlayMode component
function PlayMode() {
  const [currentSongIndex, setCurrentSongIndex] = React.useState(-1);
  const [completedSongs, setCompletedSongs] = React.useState(new Set());
  const [currentStep, setCurrentStep] = React.useState(0);
  const [songNotes, setSongNotes] = React.useState([]);
  const [showCompletionModal, setShowCompletionModal] = React.useState(false);
  const [showErrorModal, setShowErrorModal] = React.useState(false);

  const pickRandomSong = React.useCallback(() => {
    const remaining = songs.map((_, i) => i).filter(i => !completedSongs.has(i));
    if (remaining.length === 0) {
      setCompletedSongs(new Set());
      return pickRandomSong();
    }
    return remaining[Math.floor(Math.random() * remaining.length)];
  }, [completedSongs]);

  const loadSong = React.useCallback((idx) => {
    setCurrentSongIndex(idx);
    setSongNotes(songs[idx].notes.split(' '));
    setCurrentStep(0);
  }, []);
```

Chat

```
  React.useEffect(() => {
    loadSong(pickRandomSong());
  }, [loadSong, pickRandomSong]);

  const handleNotePlay = React.useCallback((note, key) => {
    synth.triggerAttack(note);

    const expected = songNotes[currentStep];
    const playedKey = key || notes.find(n => n.note === note)?.key;

    if (playedKey === expected) {
      const newStep = currentStep + 1;
      setCurrentStep(newStep);

      if (newStep === songNotes.length) {
        setShowCompletionModal(true);
      }
    } else {
      setShowErrorModal(true);
    }
  }, [currentStep, songNotes]);

  const updateProgress = React.useCallback(() => {
    return songNotes.map((k, i) => (
      <span
        key={`${i}-${k}`}
        className={`badge rounded-pill mx-1 ${i < currentStep ? 'bg-success' : 'bg-secondary'}`}
      >
        {k.toUpperCase()}
      </span>
    ));
  }, [songNotes, currentStep]);
```

```
  // FreePlay component
  function FreePlay({ freeAccounts }) {
    const [activeKeys, setActiveKeys] = React.useState(new Set());
    const [showAccounts, setShowAccounts] = React.useState(false);

  // Render the app
  const root = ReactDOM.createRoot(document.getElementById('root'));
  root.render(<App />);
</script>
```

Security Notes

Password Handling:

Stored as hash

Raw_text = for it to be not hashed when shown

SQL Injection

Session Protection: session_start() + logout feature

```
$raw_password = $_POST["password"];
$password = password_hash(password: $raw_password, algo: PASSWORD_DEFAULT);
```

Q: Why not keep it as hash for privacy purposes
A: Hard tracks example Free = 123 New = new123
if logged in as Free and clicked the see accounts or click to view function only the logged in account in this case Free will only show its password raw(unhash) the reason why we made it raw for all is it is hard to track(for us who made this project)

User/Player Journey(thisll be included in the User/Player manual txt file)

Signup -> Login -> Playmode/Freemode -> Logout

Admin Flow or whatever

Click to view

See Accounts -> Update/Delete