The Play - Checklist 3 (Advanced Features Implementation)

This in-depth checklist provides specific implementation steps to transform The Play from a basic prototype to a fully-featured event application with advanced capabilities and improved user experience.

1. User Profile System (Duration: 5 days)

1.1. Create User Profile Page (Duration: 2 days)

```
html
```

```
<!-- Create file: pages/app/profile.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Profile - The Play</title>
  <!-- Include your standard CSS and JS files here -->
  <link href="https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2.2.19/tailwind.min.css" rel="
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/
  <link rel="stylesheet" href="../../assets/styles/main.css">
  <link rel="stylesheet" href="../../assets/styles/variables.css">
  <link rel="stylesheet" href="../../assets/styles/layout.css">
  <link rel="stylesheet" href="../../assets/styles/components.css">

  <!-- Firebase Libraries -->
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-auth-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-firestore-compat.js"></script
  <script src="https://www.gstatic.com/firebasejs/9.22.2/firebase-storage-compat.js"></script>
  <script src="../../scripts/firebase/config.js"></script>
</head>
<body class="bg-gray-100">
  <!-- Header Container -->
  <div id="header-container"></div>

  <!-- Main Content -->
  <div class="container mx-auto px-4 py-8">
    <div class="max-w-4xl mx-auto">
      <!-- Profile Header -->
      <div class="bg-white rounded-xl shadow-md overflow-hidden mb-6">
        <div class="bg-gradient-to-r from-purple-600 to-pink-500 h-48 relative">
          <!-- Edit Cover Button -->
          <button id="edit-cover" class="absolute top-4 right-4 bg-white bg-opacity-25 p-2 rour
            <i class="fas fa-camera text-white"></i>
          </button>

          <!-- Profile Picture -->
          <div class="absolute -bottom-16 left-8">
            <div class="w-32 h-32 rounded-full border-4 border-white bg-gray-200 overflow-hidde
              <img id="user-profile-pic" src="/api/placeholder/128/128" alt="Profile" class="w-
              <button id="edit-profile-pic" class="absolute bottom-0 inset-x-0 h-10 bg-black bg
                <i class="fas fa-camera mr-2"></i> Change
```

```
        </button>
      </div>
    </div>
  </div>

  <div class="p-6 pt-20">
    <div class="flex justify-between items-start">
      <div>
        <h1 id="user-display-name" class="text-2xl font-bold">User Name</h1>
        <p id="user-email" class="text-gray-600">user@example.com</p>
        <p id="user-join-date" class="text-sm text-gray-500 mt-1">Joined May 2025</p>
      </div>
      <button id="edit-profile" class="btn-secondary text-sm">
        <i class="fas fa-edit mr-2"></i> Edit Profile
      </button>
    </div>

    <div class="flex items-center mt-6 space-x-8">
      <div class="text-center">
        <p class="text-xl font-bold" id="count-events">0</p>
        <p class="text-gray-600 text-sm">Events Attended</p>
      </div>
      <div class="text-center">
        <p class="text-xl font-bold" id="count-bookmarks">0</p>
        <p class="text-gray-600 text-sm">Bookmarked</p>
      </div>
      <div class="text-center">
        <p class="text-xl font-bold" id="count-vibes">0</p>
        <p class="text-gray-600 text-sm">Vibe Checks</p>
      </div>
    </div>
  </div>
</div>

<!-- Tabs Navigation -->
<div class="flex border-b border-gray-200 mb-6">
  <button class="py-2 px-4 tab-active profile-tab" data-tab="bookmarked">Bookmarked</butt
  <button class="py-2 px-4 text-gray-600 profile-tab" data-tab="attending">Attending</but
  <button class="py-2 px-4 text-gray-600 profile-tab" data-tab="past">Past Events</buttor
  <button class="py-2 px-4 text-gray-600 profile-tab" data-tab="vibes">Vibe Checks</buttc
</div>

<!-- Tabs Content -->
<div id="tab-bookmarked" class="tab-content">
```

```
      <div id="bookmarked-events" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6
        <!-- Bookmarked events will be loaded here -->
        <div class="text-center py-12 col-span-full hidden" id="no-bookmarks">
          <p class="text-gray-500">You haven't bookmarked any events yet.</p>
          <a href="index.html" class="btn-primary inline-block mt-4">Discover Events</a>
        </div>
      </div>
    </div>

    <div id="tab-attending" class="tab-content hidden">
      <div id="attending-events" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"
        <!-- Attending events will be loaded here -->
        <div class="text-center py-12 col-span-full hidden" id="no-attending">
          <p class="text-gray-500">You haven't RSVP'd to any upcoming events.</p>
          <a href="index.html" class="btn-primary inline-block mt-4">Discover Events</a>
        </div>
      </div>
    </div>

    <div id="tab-past" class="tab-content hidden">
      <div id="past-events" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
        <!-- Past events will be loaded here -->
        <div class="text-center py-12 col-span-full hidden" id="no-past">
          <p class="text-gray-500">You haven't attended any events yet.</p>
        </div>
      </div>
    </div>

    <div id="tab-vibes" class="tab-content hidden">
      <div id="user-vibes" class="grid grid-cols-1 md:grid-cols-2 gap-6">
        <!-- Vibe checks will be loaded here -->
        <div class="text-center py-12 col-span-full hidden" id="no-vibes">
          <p class="text-gray-500">You haven't posted any vibe checks yet.</p>
          <button class="btn-primary inline-block mt-4 open-vibe-modal">Post a Vibe Check</bu
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Edit Profile Modal -->
<div id="modal-container"></div>

<!-- Scripts -->
```

```
<script type="module">
  import { loadComponent } from '../../scripts/utils/component-loader.js';
  import userProfile from '../../scripts/components/profile.js';
  import errorHandler from '../../scripts/utils/error-handler.js';

  document.addEventListener('DOMContentLoaded', function() {
    // Load header component
    loadComponent('header-container', '../../components/header.html');

    // Load the edit profile modal
    loadComponent('modal-container', '../../components/modals/edit-profile-modal.html');

    // Initialize user profile
    initializeProfile();

    // Set up tab switching
    const tabs = document.querySelectorAll('.profile-tab');
    tabs.forEach(tab => {
      tab.addEventListener('click', function() {
        // Update tab styling
        tabs.forEach(t => {
          t.classList.remove('tab-active');
          t.classList.add('text-gray-600');
        });
        tab.classList.add('tab-active');
        tab.classList.remove('text-gray-600');

        // Show selected tab content
        const tabName = tab.getAttribute('data-tab');
        document.querySelectorAll('.tab-content').forEach(content => {
          content.classList.add('hidden');
        });
        document.getElementById(`tab-${tabName}`).classList.remove('hidden');

        // Load content for the selected tab
        loadTabContent(tabName);
      });
    });

    // Set up open vibe modal button
    document.querySelectorAll('.open-vibe-modal').forEach(btn => {
      btn.addEventListener('click', function() {
        import('../../scripts/components/modals.js')
          .then(module => {
```

```
          module.openModal('vibeCheckModal');
        });
    });
  });
});

async function initializeProfile() {
  try {
    // Check authentication
    const user = firebase.auth().currentUser;
    if (!user) {
      // Wait for auth state to be determined
      firebase.auth().onAuthStateChanged(user => {
        if (user) {
          loadUserProfile(user);
        } else {
          window.location.href = '../auth/login.html';
        }
      });
    } else {
      loadUserProfile(user);
    }
  } catch (error) {
    errorHandler.showError('Failed to load profile. Please try again later.');
    console.error('Profile initialization error:', error);
  }
}

async function loadUserProfile(user) {
  try {
    // Load user data
    await userProfile.initialize();

    // Update profile UI
    document.getElementById('user-display-name').textContent = userProfile.getDisplayName()
    document.getElementById('user-email').textContent = user.email;

    // Set profile pic
    const profilePic = document.getElementById('user-profile-pic');
    profilePic.src = userProfile.getPhotoURL();

    // Set join date
    if (userProfile.userData && userProfile.userData.createdAt) {
      const joinDate = new Date(userProfile.userData.createdAt.toDate());
```

```
      const options = { year: 'numeric', month: 'long' };
      document.getElementById('user-join-date').textContent = `Joined ${joinDate.toLocaleDa
    }

    // Set up profile pic change
    document.getElementById('edit-profile-pic').addEventListener('click', changeProfilePict

    // Set up cover change
    document.getElementById('edit-cover').addEventListener('click', changeCoverPhoto);

    // Set up edit profile button
    document.getElementById('edit-profile').addEventListener('click', function() {
      import('../../scripts/components/modals.js')
        .then(module => {
          module.openModal('editProfileModal');
        });
    });

    // Load initial tab (bookmarked)
    loadTabContent('bookmarked');

    // Load counts
    loadUserCounts();
  } catch (error) {
    errorHandler.showError('Failed to load profile data.');
    console.error('Error loading user profile:', error);
  }
}

async function loadUserCounts() {
  const db = firebase.firestore();
  const user = firebase.auth().currentUser;

  if (!user) return;

  try {
    // Get bookmarks count
    const bookmarksSnapshot = await db.collection('bookmarks')
      .where('userId', '==', user.uid)
      .get();
    document.getElementById('count-bookmarks').textContent = bookmarksSnapshot.size;

    // Get attending count (upcoming events)
    const now = new Date();
```

```
    const attendingSnapshot = await db.collection('attendees')
      .where('userId', '==', user.uid)
      .where('eventDate', '>=', now)
      .get();
    document.getElementById('count-events').textContent = attendingSnapshot.size;

    // Get vibe checks count
    const vibesSnapshot = await db.collection('vibeChecks')
      .where('userId', '==', user.uid)
      .get();
    document.getElementById('count-vibes').textContent = vibesSnapshot.size;
  } catch (error) {
    console.error('Error loading user counts:', error);
  }
}

async function loadTabContent(tabName) {
  const db = firebase.firestore();
  const user = firebase.auth().currentUser;

  if (!user) return;

  try {
    switch (tabName) {
      case 'bookmarked':
        // Load bookmarked events
        const bookmarks = await loadBookmarkedEvents();
        if (bookmarks.length === 0) {
          document.getElementById('no-bookmarks').classList.remove('hidden');
        } else {
          document.getElementById('no-bookmarks').classList.add('hidden');
        }
        break;

      case 'attending':
        // Load events user is attending
        const attending = await loadAttendingEvents();
        if (attending.length === 0) {
          document.getElementById('no-attending').classList.remove('hidden');
        } else {
          document.getElementById('no-attending').classList.add('hidden');
        }
        break;
```

```
      case 'past':
        // Load past events
        const past = await loadPastEvents();
        if (past.length === 0) {
          document.getElementById('no-past').classList.remove('hidden');
        } else {
          document.getElementById('no-past').classList.add('hidden');
        }
        break;

      case 'vibes':
        // Load user's vibe checks
        const vibes = await loadUserVibes();
        if (vibes.length === 0) {
          document.getElementById('no-vibes').classList.remove('hidden');
        } else {
          document.getElementById('no-vibes').classList.add('hidden');
        }
        break;
    }
  } catch (error) {
    console.error(`Error loading ${tabName} content:`, error);
    errorHandler.showToast(`Failed to load ${tabName} content.`, 'error');
  }
}

async function loadBookmarkedEvents() {
  const containerId = 'bookmarked-events';
  const container = document.getElementById(containerId);
  container.innerHTML = '';

  try {
    // Get user bookmarks
    const bookmarks = await userProfile.getBookmarkedEvents();

    // Import event card creator and display events
    const eventsModule = await import('../../scripts/components/events.js');
    bookmarks.forEach(event => {
      eventsModule.createEventCard(event, containerId);
    });

    return bookmarks;
  } catch (error) {
    console.error('Error loading bookmarked events:', error);
```

```
    return [];
  }
}

async function loadAttendingEvents() {
  // Similar implementation to loadBookmarkedEvents
  // This would load events the user has RSVP'd to
  return [];
}

async function loadPastEvents() {
  // Load past events the user has attended
  return [];
}

async function loadUserVibes() {
  // Load vibe checks posted by the user
  return [];
}

function changeProfilePicture() {
  const fileInput = document.createElement('input');
  fileInput.type = 'file';
  fileInput.accept = 'image/*';

  fileInput.addEventListener('change', async (event) => {
    if (event.target.files.length > 0) {
      const file = event.target.files[0];

      try {
        // Show loading state
        errorHandler.showToast('Uploading profile picture...', 'info');

        // Upload to Firebase Storage
        const storageRef = firebase.storage().ref();
        const fileRef = storageRef.child(`profile-pics/${firebase.auth().currentUser.uid}`)

        await fileRef.put(file);
        const downloadURL = await fileRef.getDownloadURL();

        // Update user profile
        await firebase.auth().currentUser.updateProfile({
          photoURL: downloadURL
        });
```

```
            // Update in Firestore
            await userProfile.updateProfile({
              photoURL: downloadURL
            });

            // Update UI
            document.getElementById('user-profile-pic').src = downloadURL;

            errorHandler.showToast('Profile picture updated!', 'success');
          } catch (error) {
            console.error('Error uploading profile picture:', error);
            errorHandler.showToast('Failed to update profile picture.', 'error');
          }
        }
      });

      fileInput.click();
    }

    function changeCoverPhoto() {
      // Similar to changeProfilePicture but for cover photo
      alert('Cover photo upload will be implemented in a future update');
    }
  </script>
</body>
</html>
```

1.2. Create Edit Profile Modal (Duration: 1 day)

```
html
```

```
<!-- Create file: components/modals/edit-profile-modal.html -->
<div class="fixed inset-0 bg-black bg-opacity-50 hidden flex items-center justify-center z-50"
  <div class="bg-white rounded-xl w-11/12 max-w-md p-6" onclick="event.stopPropagation();">
    <div class="flex justify-between items-center mb-4">
      <h3 class="text-xl font-bold">Edit Profile</h3>
      <button class="text-gray-500 hover:text-gray-700 close-modal-btn">
        <i class="fas fa-times"></i>
      </button>
    </div>

    <form id="edit-profile-form">
      <div class="mb-4">
        <label for="display-name" class="block text-gray-700 text-sm font-medium mb-2">Display
        <input type="text" id="display-name" class="w-full p-3 border border-gray-300 rounded-l
      </div>

      <div class="mb-4">
        <label for="bio" class="block text-gray-700 text-sm font-medium mb-2">Bio</label>
        <textarea id="bio" class="w-full p-3 border border-gray-300 rounded-lg h-24"></textarea
      </div>

      <div class="mb-4">
        <label for="location" class="block text-gray-700 text-sm font-medium mb-2">Location</la
        <input type="text" id="location" class="w-full p-3 border border-gray-300 rounded-lg">
      </div>

      <div class="mb-4">
        <label for="instagram" class="block text-gray-700 text-sm font-medium mb-2">Instagram U
        <div class="flex items-center">
          <span class="text-gray-500 mr-2">@</span>
          <input type="text" id="instagram" class="flex-1 p-3 border border-gray-300 rounded-lg
        </div>
      </div>

      <div class="flex justify-end">
        <button type="button" class="mr-4 px-6 py-3 rounded-lg border border-gray-300 text-gray
        <button type="submit" class="btn-primary px-6 py-3">Save Changes</button>
      </div>
    </form>
  </div>
</div>

<script>
```

```
// Close modal functionality
document.querySelectorAll('#editProfileModal .close-modal-btn').forEach(btn => {
  btn.addEventListener('click', function() {
    const modal = document.getElementById('editProfileModal');
    modal.classList.add('hidden');
    document.body.style.overflow = 'auto';
  });
});

// Close when clicking outside
document.getElementById('editProfileModal').addEventListener('click', function(e) {
  if (e.target === this) {
    this.classList.add('hidden');
    document.body.style.overflow = 'auto';
  }
});

// Initialize form with user data
document.addEventListener('DOMContentLoaded', function() {
  import('../../scripts/components/profile.js')
    .then(module => {
      const userProfile = module.default;

      // Load user profile data
      userProfile.initialize().then(() => {
        if (userProfile.userData) {
          document.getElementById('display-name').value = userProfile.userData.displayName ||
          document.getElementById('bio').value = userProfile.userData.bio || '';
          document.getElementById('location').value = userProfile.userData.location || '';
          document.getElementById('instagram').value = userProfile.userData.instagram || '';
        }
      });
    });

  // Form submission
  document.getElementById('edit-profile-form').addEventListener('submit', async function(e) {
    e.preventDefault();

    const module = await import('../../scripts/components/profile.js');
    const userProfile = module.default;
    const errorHandler = (await import('../../scripts/utils/error-handler.js')).default;

    try {
      // Show loading state
```

```
    const submitBtn = this.querySelector('button[type="submit"]');
    submitBtn.disabled = true;
    submitBtn.innerHTML = '<i class="fas fa-spinner fa-spin mr-2"></i> Saving...';

    // Get form values
    const displayName = document.getElementById('display-name').value.trim();
    const bio = document.getElementById('bio').value.trim();
    const location = document.getElementById('location').value.trim();
    const instagram = document.getElementById('instagram').value.trim();

    // Validate display name
    if (!displayName) {
      errorHandler.showToast('Display name is required', 'error');
      submitBtn.disabled = false;
      submitBtn.innerHTML = 'Save Changes';
      return;
    }

    // Update profile
    await userProfile.updateProfile({
      displayName,
      bio,
      location,
      instagram
    });

    // Update display name in Firebase Auth
    await firebase.auth().currentUser.updateProfile({
      displayName: displayName
    });

    // Update UI
    document.getElementById('user-display-name').textContent = displayName;

    // Close modal
    document.getElementById('editProfileModal').classList.add('hidden');
    document.body.style.overflow = 'auto';

    // Show success message
    errorHandler.showToast('Profile updated successfully', 'success');
  } catch (error) {
    console.error('Error updating profile:', error);
    errorHandler.showToast('Failed to update profile', 'error');
  } finally {
```

```
        // Reset button
        const submitBtn = this.querySelector('button[type="submit"]');
        submitBtn.disabled = false;
        submitBtn.innerHTML = 'Save Changes';
      }
    });
  });
</script>
```

1.3. Enhance User Profile Module (Duration: 2 days)

```
javascript
```

```
// Enhance file: scripts/components/profile.js
class UserProfile {
  constructor() {
    this.currentUser = null;
    this.userData = null;
    this.isInitialized = false;
    this.profileListeners = [];
  }

  async initialize() {
    if (this.isInitialized) return;

    // Wait for Firebase Auth to initialize
    if (!firebase.auth().currentUser) {
      await new Promise(resolve => {
        const unsubscribe = firebase.auth().onAuthStateChanged(user => {
          this.currentUser = user;
          unsubscribe();
          resolve();
        });
      });
    } else {
      this.currentUser = firebase.auth().currentUser;
    }

    if (this.currentUser) {
      await this.loadUserData();
      this.setupRealTimeUpdates();
    }

    this.isInitialized = true;
    return this.userData;
  }

  setupRealTimeUpdates() {
    // Remove any existing listeners
    this.profileListeners.forEach(unsubscribe => unsubscribe());
    this.profileListeners = [];

    // Set up listener for user profile changes
    const db = firebase.firestore();
    const userDocRef = db.collection('users').doc(this.currentUser.uid);
```

```
  const unsubscribe = userDocRef.onSnapshot(doc => {
    if (doc.exists) {
      this.userData = doc.data();
      // Notify UI components that might need updating
      this.notifyProfileUpdate();
    }
  }, error => {
    console.error('Error in user profile snapshot listener:', error);
  });

  this.profileListeners.push(unsubscribe);
}

notifyProfileUpdate() {
  // Custom event to notify UI components of profile updates
  const event = new CustomEvent('user-profile-updated', {
    detail: {
      userData: this.userData
    }
  });
  document.dispatchEvent(event);
}

async loadUserData() {
  if (!this.currentUser) return null;

  try {
    const db = firebase.firestore();
    const doc = await db.collection('users').doc(this.currentUser.uid).get();

    if (doc.exists) {
      this.userData = doc.data();
      return this.userData;
    } else {
      // Create user profile if it doesn't exist
      const userData = {
        email: this.currentUser.email,
        displayName: this.currentUser.displayName || this.currentUser.email.split('@')[0],
        photoURL: this.currentUser.photoURL || null,
        createdAt: new Date(),
        lastLogin: new Date(),
        bio: '',
        location: '',
        instagram: '',
```

```
        preferences: {
          eventTypes: [],
          notificationsEnabled: true
        }
      };

      await db.collection('users').doc(this.currentUser.uid).set(userData);
      this.userData = userData;
      return userData;
    }
  } catch (error) {
    console.error('Error loading user data:', error);
    return null;
  }
}

getDisplayName() {
  if (this.userData && this.userData.displayName) {
    return this.userData.displayName;
  }

  if (this.currentUser) {
    return this.currentUser.displayName || this.currentUser.email.split('@')[0];
  }

  return 'User';
}

getPhotoURL() {
  if (this.userData && this.userData.photoURL) {
    return this.userData.photoURL;
  }

  if (this.currentUser && this.currentUser.photoURL) {
    return this.currentUser.photoURL;
  }

  return '/api/placeholder/100/100';
}

async updateProfile(data) {
  if (!this.currentUser) return false;

  try {
```

```
    const db = firebase.firestore();
    await db.collection('users').doc(this.currentUser.uid).update({
      ...data,
      updatedAt: new Date()
    });

    // Refresh user data
    return await this.loadUserData();
  } catch (error) {
    console.error('Error updating profile:', error);
    throw error;
  }
}

async getBookmarkedEvents() {
  if (!this.currentUser) return [];

  try {
    const db = firebase.firestore();
    const snapshot = await db.collection('bookmarks')
      .where('userId', '==', this.currentUser.uid)
      .get();

    const bookmarkIds = snapshot.docs.map(doc => doc.data().eventId);

    if (bookmarkIds.length === 0) return [];

    // Get the actual events
    const eventsSnapshot = await db.collection('events')
      .where(firebase.firestore.FieldPath.documentId(), 'in', bookmarkIds)
      .get();

    return eventsSnapshot.docs.map(doc => {
      return {
        id: doc.id,
        ...doc.data()
      };
    });
  } catch (error) {
    console.error('Error getting bookmarked events:', error);
    return [];
  }
}
```

```
async getAttendingEvents() {
  if (!this.currentUser) return [];

  try {
    const db = firebase.firestore();
    const now = new Date();

    // Get events the user is attending that haven't happened yet
    const snapshot = await db.collection('attendees')
      .where('userId', '==', this.currentUser.uid)
      .where('eventDate', '>=', now)
      .get();

    const eventIds = snapshot.docs.map(doc => doc.data().eventId);

    if (eventIds.length === 0) return [];

    // Get the actual events
    const eventsSnapshot = await db.collection('events')
      .where(firebase.firestore.FieldPath.documentId(), 'in', eventIds)
      .get();

    return eventsSnapshot.docs.map(doc => {
      return {
        id: doc.id,
        ...doc.data()
      };
    });
  } catch (error) {
    console.error('Error getting attending events:', error);
    return [];
  }
}

async getPastEvents() {
  if (!this.currentUser) return [];

  try {
    const db = firebase.firestore();
    const now = new Date();

    // Get events the user attended that have already happened
    const snapshot = await db.collection('attendees')
      .where('userId', '==', this.currentUser.uid)
```

```
  .where('eventDate', '<', now)
  .get();

const eventIds = snapshot.docs.map(doc => doc.data().eventId);

if (eventIds.length === 0) return [];

// Get the actual events
const eventsSnapshot = await db.collection('events')
  .where(firebase.firestore.FieldPath.documentId(), 'in',
```