

FIT5032 Assessed Lab 8 Submission

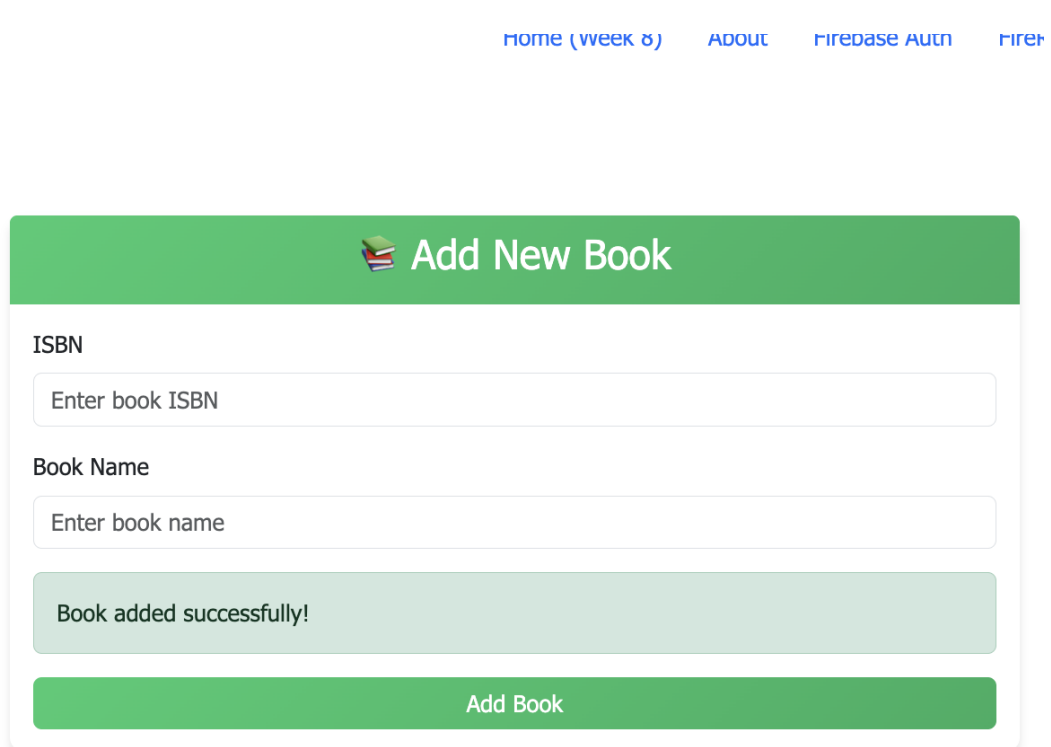
Firestore Database Integration

Student Name: [Du Daoan]
Student ID: [35523166]

1 EFOLIO TASK 8.1 - Basic Firestore Integration

1.1 Screenshot Set 1: Add Book Implementation

1.1.1 Browser View



home (week 8) ABOUT Firestore AUTH Firestore

Add New Book

ISBN

Book Name

Book added successfully!

Add Book

Figure 1: AddBookView.vue page showing the form and book list

Required: Screenshot showing the AddBook page with form and book list components.

1.1.2 Visual Studio Code Implementation

Required: Screenshot showing the AddBookView.vue code implementation.

1.2 Screenshot Set 2: Firestore Database

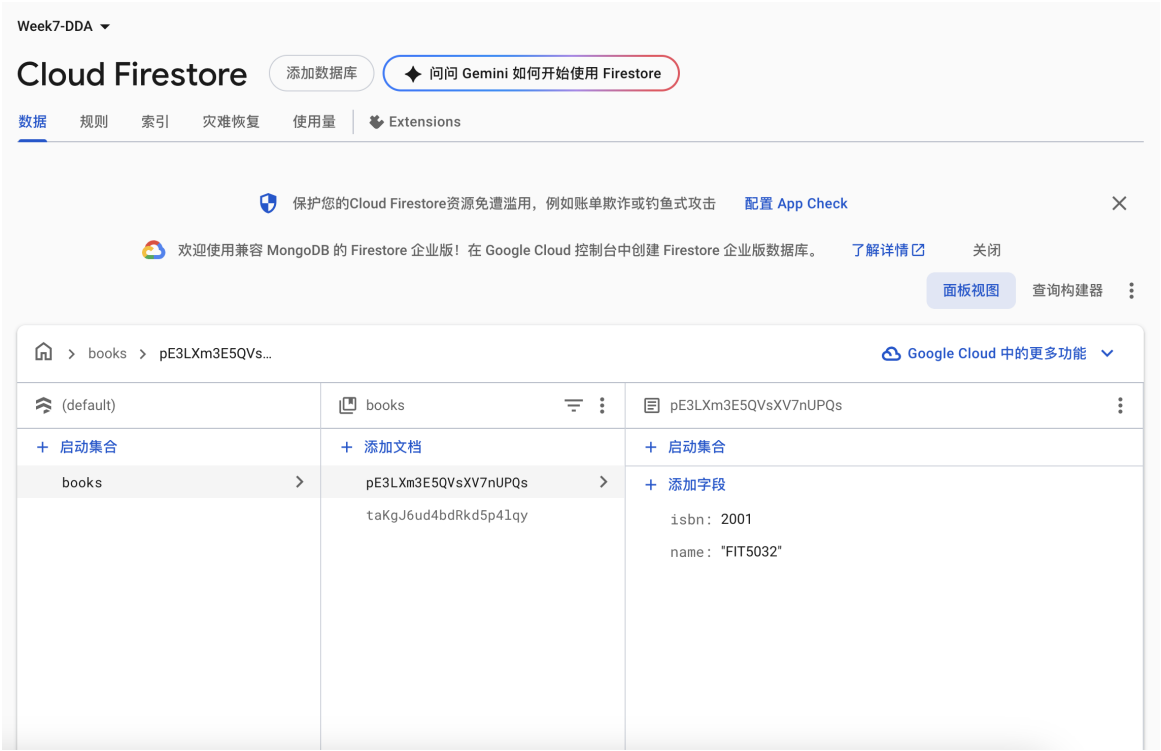


Figure 2: Firestore console showing added book data

Required: Screenshot of Firestore console showing the books collection with added data.

2 EFOLIO TASK 8.2 - Advanced Firestore Operations

2.1 Screenshot Set 1: Update and Delete Operations

2.1.1 Update Operation

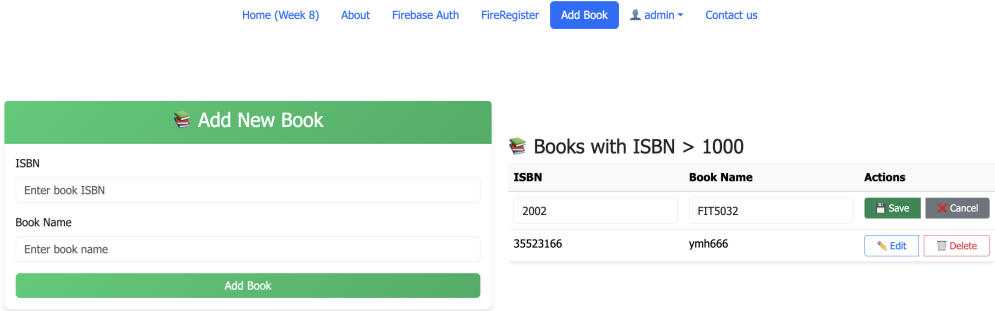


Figure 3: Book update functionality in browser

```

src > components > BookList.vue > {} script setup
 92  <script setup>
105  const fetchBooks = async () => {
127    loading.value = false
128  }
129  }
130
131  // Edit functions
132  const startEdit = (book) => {
133    editingBook.value = { ...book }
134  }
135
136  const cancelEdit = () => {
137    editingBook.value = null
138  }
139  | %%L to chat, %%K to generate
140  const saveEdit = async () => {
141    try {
142      if (!editingBook.value) return
143
144      const bookRef = doc(db, "books", editingBook.value.id)
145      await updateDoc(bookRef, {
146        isbn: editingBook.value.isbn,
147        name: editingBook.value.name
148      })
149
150      console.log('🔥 Book updated successfully')
151      await fetchBooks()
152      editingBook.value = null
153    } catch (err) {
154      console.error('Error updating book:', err)
155      error.value = 'Failed to update book: ' + err.message
156    }
157  }
158
159  // Delete functions
160  const confirmDelete = (book) => {
161    bookToDelete.value = book
162    const modal = new Modal(deleteModal.value)
163    modal.show()
164  }
165

```

Figure 4: Update implementation in VS Code

2.1.2 Delete Operation

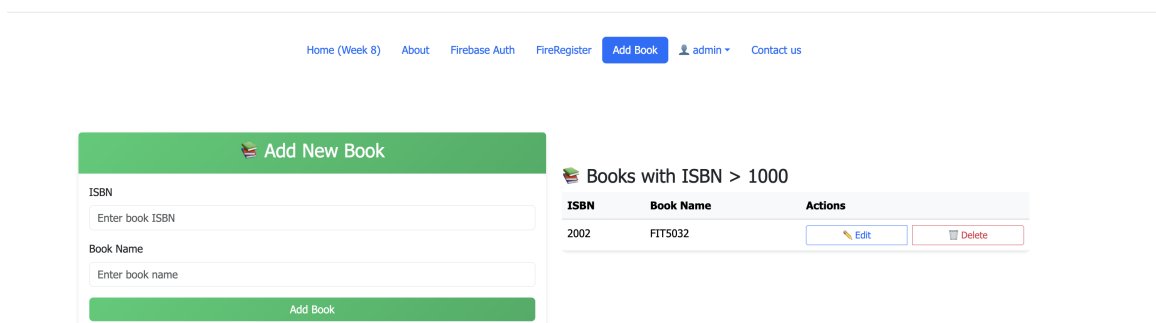


Figure 5: Book deletion functionality in browser

```
src > components > BookList.vue > {} script setup
192 <script setup>
193   const deleteBook = async () => {
194   }
195 }
196
197 // Delete functions
198 const confirmDelete = (book) => {
199   bookToDelete.value = book
200   const modal = new Modal(deleteModal.value)
201   modal.show()
202 }
203
204 const deleteBook = async () => {
205   try {
206     if (!bookToDelete.value) return
207
208     const bookRef = doc(db, "books", bookToDelete.value.id)
209     await deleteDoc(bookRef)
210
211     console.log('🔥 Book deleted successfully')
212     const modal = Modal.getInstance(deleteModal.value)
213     modal.hide()
214     await fetchBooks()
215     bookToDelete.value = null
216   } catch (err) {
217     console.error('Error deleting book:', err)
218     error.value = 'Failed to delete book: ' + err.message
219   }
220 }
221
222 // Fetch books when component is mounted
223 onMounted(fetchBooks)
224
225 // Expose the fetchBooks method to parent components
226 defineExpose({ fetchBooks })
227 </script>
228
229 <style scoped>
230 .table {
231   background-color: white;
232   border-radius: 8px;
233   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
234 }
235 </style>
```

Figure 6: Delete implementation in VS Code

2.2 Screenshot Set 2: Advanced Queries

2.2.1 Query Implementation

```
const fetchBooks = async () => {
  try {
    loading.value = true
    error.value = null

    // Build query constraints
    const constraints = []

    // ISBN range filter
    if (queryParams.value.isbnRange === 'over1000') {
      constraints.push(where('isbn', '>', 1000))
    } else if (queryParams.value.isbnRange === 'over5000') {
      constraints.push(where('isbn', '>', 5000))
    }

    // Sort order
    constraints.push(orderBy(
      queryParams.value.sortBy,
      queryParams.value.sortDir
    ))

    // Result limit
    if (queryParams.value.limit) {
      constraints.push(limit(queryParams.value.limit))
    }

    // Create and execute query
    console.log('🔍 Executing query with constraints:', constraints)
    const q = query(collection(db, "books"), ...constraints)
    const querySnapshot = await getDocs(q)

    books.value = querySnapshot.docs.map(doc => ({
      id: doc.id,
```

Figure 7: Implementation of where, orderBy, and limit queries

2.2.2 Query Results

The screenshot displays a web application interface with a navigation bar at the top containing links: Home (Week 8), About, Firebase Auth, FireRegister, Add Book, admin, and Contact us. Below the navigation bar, there are two main sections. On the left is the 'Add New Book' form, which includes input fields for 'ISBN' (with placeholder 'Enter book ISBN') and 'Book Name' (with placeholder 'Enter book name'), and a green 'Add Book' button. On the right is the 'Advanced Book Search' form, which includes dropdown menus for 'ISBN Range' (set to '> 1000'), 'Sort By' (set to 'ISBN'), 'Sort Direction' (set to 'Ascending'), and 'Limit Results' (set to '5 Books'). Below these dropdowns is a blue 'Search Books' button. Below the search form, a light blue banner states 'Found 2 books matching your criteria'. Below this banner is a table with the following data:

ISBN	Book Name	Actions
2002	FIT5032	Edit Delete
3552	ymh	Edit Delete

Figure 8: Query results displayed in browser

3 Technical Achievement Summary

This implementation demonstrates:

- **Firestore Integration:** Complete setup with Vue.js
- **CRUD Operations:** Create, Read, Update, Delete functionality
- **Advanced Queries:** Using where, orderBy, and limit
- **Real-time Updates:** Automatic UI updates on data changes
- **Component Architecture:** Reusable BookList component
- **Form Validation:** Required fields and type checking
- **Error Handling:** Proper error messages and loading states