**✅ Week 1: Hibernate Basics + Mini Project: Student Management System**

**🔧 Project Name: StudentManagementSystem**

📌 Description: Build a system to add, view, update, and delete student records using Hibernate basics.

| **Day** | **Focus Area** | **Tasks** |
| --- | --- | --- |
| **Day 1** | Intro to ORM + Hibernate vs JDBC | Learn what ORM is, why Hibernate is better than JDBC, and the big picture (architecture) |
| **Day 2** | Hibernate Setup | Create a Maven project, add Hibernate & MySQL dependencies, create Hibernate config file (XML), and test DB connection |
| **Day 3** | Entity Class & SessionFactory | Create Student.java entity, configure Hibernate with annotations, build HibernateUtil.java |
| **Day 4** | Add Student Feature | Use session.save() to add student to DB |
| **Day 5** | View & Update Student | Implement get() and update() methods |
| **Day 6** | Delete & List All Students | Implement delete() and HQL to list all students |
| **Day 7** | Testing + Clean Code | Refactor code, test all CRUD operations, and finalize the project |

**✅ Week 2: Hibernate Mapping + Mini Project: Employee Management System**

**🔧 Project Name: EmployeeManagementSystem**

📌 Description: Build a system with advanced Hibernate mappings like One-to-One, One-to-Many, and use of HQL.

| **Day** | **Focus Area** | **Tasks** |
| --- | --- | --- |
| **Day 1** | Hibernate Mapping Overview | Learn about .hbm.xml, annotations, and entity relationships |
| **Day 2** | One-to-One Mapping | Add an Address entity and map it to Employee using @OneToOne |
| **Day 3** | One-to-Many Mapping | Add a Project entity – one employee can work on multiple projects |
| **Day 4** | CRUD with Mappings | Add, update, and view employees along with their address and project info |
| **Day 5** | HQL Queries | Use HQL for custom queries (e.g., employees by department or project count) |
| **Day 6** | Lazy vs Eager Loading | Experiment with loading strategies and observe DB behavior |
| **Day 7** | Finalize + Testing | Code cleanup, handle exceptions, and test the complete project |

**✅ Week 3: Hibernate Queries & Annotations**

**🔧 Project Name: LibraryQuerySystem**

📌 **Goal**: Build a system to manage books, authors, and search functionality using HQL, Criteria API, and annotations.

| **Day** | **Focus Area** | **Tasks** |
| --- | --- | --- |
| **Day 1** | HQL Basics | Write simple HQL queries to fetch all books and filter by title/author |
| **Day 2** | HQL Advanced | Use setParameter(), pagination (setMaxResults()), and ordering |
| **Day 3** | Criteria API (old) | Use Criteria API with Restrictions: eq(), like(), gt(), etc. |
| **Day 4** | Criteria API (new - JPA) | Implement using CriteriaBuilder, CriteriaQuery, Root |
| **Day 5** | Entity Annotations | Refactor code to use annotations (@Entity, @Table, @Id, etc.) |
| **Day 6** | Transaction Management | Use @Transactional, handle rollback, exception propagation |
| **Day 7** | Testing + Wrap-up | Final testing, exception handling, refactor reusable code |

**✅ Week 4: Mappings, Caching & Advanced Session Management**

**🔧 Project Name: CourseEnrollmentSystem**

📌 **Goal**: Create a complete system to enroll students in courses, manage relationships, use caching, and apply lazy/eager loading.

| **Day** | **Focus Area** | **Tasks** |
| --- | --- | --- |
| **Day 1** | One-to-One & One-to-Many Mapping | Map Student ↔ Profile, and Course ↔ Instructor |
| **Day 2** | Many-to-Many Mapping | Map Student ↔ Course with join table |
| **Day 3** | Lazy vs Eager Loading | Annotate and observe behavior using logs/queries |
| **Day 4** | First-Level Cache | Explore Hibernate’s default session cache |
| **Day 5** | Second-Level Cache | Enable EHCache or Caffeine and test across sessions |
| **Day 6** | save() vs saveOrUpdate(), get() vs load(), merge() | Implement examples for each method with logs |
| **Day 7** | Final Testing + Optimization | Run full test scenarios, finalize code, and comment/document |