ProjectsApp.java class

```
// @formatter:off
private List<String> operations = List.of(
    "1) Add a project",
    "2) List projects",
    "3) Select a project",
    "4) Update project details",
    "5) Delete a project"
);
// @formatter:on
private void processUserSelections() {
  boolean done = false;

  while(!done) {
    try {
      int selection = getUserSelection();

      switch(selection) {
        case -1:
          done = exitMenu();
          break;

        case 1:
          createProject();
          break;

        case 2:
          listProjects();
          break;

        case 3:
          selectProject();
          break;

        case 4:
          updateProjectDetails();
          break;

        case 5:
          deleteProject();
          break;

        default:
          System.out.println("\n" + selection + " is not a valid selection. Try again.");
          break;
      }
    }
    catch(Exception e) {
      System.out.println("\nError: " + e + " Try again.");
    }
  }
}
```

```
private void deleteProject() {
  listProjects();

  Integer projectId = getIntInput("Enter the ID of the project to delete");

  projectService.deleteProject(projectId);
  System.out.println("Project " + projectId + " was deleted successfully.");

  if(Objects.nonNull(curProject) && curProject.getProjectId().equals(projectId)) {
    curProject = null;
  }
}
```

```
private void updateProjectDetails() {
  if(Objects.isNull(curProject)) {
    System.out.println("\nPlease select a project.");
    return;
  }

  String projectName =
      getStringInput("Enter the project name [" + curProject.getProjectName() + "]");

  BigDecimal estimatedHours =
      getDecimalInput("Enter the estimated hours [" + curProject.getEstimatedHours() + "]");

  BigDecimal actualHours =
      getDecimalInput("Enter the actual hours + [" + curProject.getActualHours() + "]");

  Integer difficulty =
      getIntInput("Enter the project difficulty (1-5) [" + curProject.getDifficulty() + "]");

  String notes = getStringInput("Enter the project notes [" + curProject.getNotes() + "]");

  Project project = new Project();

  project.setProjectId(curProject.getProjectId());
  project.setProjectName(Objects.isNull(projectName) ? curProject.getProjectName() : projectName);

  project.setEstimatedHours(
      Objects.isNull(estimatedHours) ? curProject.getEstimatedHours() : estimatedHours);

  project.setActualHours(Objects.isNull(actualHours) ? curProject.getActualHours() : actualHours);
  project.setDifficulty(Objects.isNull(difficulty) ? curProject.getDifficulty() : difficulty);
  project.setNotes(Objects.isNull(notes) ? curProject.getNotes() : notes);

  projectService.modifyProjectDetails(project);

  curProject = projectService.fetchProjectById(curProject.getProjectId());
}
```

ProjectService.java class

```
public void modifyProjectDetails(Project project) {
  if(!projectDao.modifyProjectDetails(project)) {
    throw new DbException("Project with ID=" + project.getProjectId() + " does not exist.");
  }
}

/**
 * @param projectId
 */
public void deleteProject(Integer projectId) {
  if(!projectDao.deleteProject(projectId)) {
    throw new DbException("Project with ID=" + projectId + " does not exist.");
  }
}
```

ProjectDao.java class

```java
public boolean modifyProjectDetails(Project project) {
  // @formatter:off
  String sql = ""
      + "UPDATE " + PROJECT_TABLE + " SET "
      + "project_name = ?, "
      + "estimated_hours = ?, "
      + "actual_hours = ?, "
      + "difficulty = ?, "
      + "notes = ? "
      + "WHERE project_id = ?";
  // @formatter:on

  try(Connection conn = DbConnection.getConnection()) {
    startTransaction(conn);

    try(PreparedStatement stmt = conn.prepareStatement(sql)) {
      setParameter(stmt, 1, project.getProjectName(), String.class);
      setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
      setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
      setParameter(stmt, 4, project.getDifficulty(), Integer.class);
      setParameter(stmt, 5, project.getNotes(), String.class);
      setParameter(stmt, 6, project.getProjectId(), Integer.class);

      boolean modified = stmt.executeUpdate() == 1;
      commitTransaction(conn);

      return modified;
    }
    catch(Exception e) {
      rollbackTransaction(conn);
      throw new DbException(e);
    }
  }
  catch(SQLException e) {
    throw new DbException(e);
  }
}
```

```java
public boolean deleteProject(Integer projectId) {
  String sql = "DELETE FROM " + PROJECT_TABLE + " WHERE project_id = ?";

  try(Connection conn = DbConnection.getConnection()) {
    startTransaction(conn);

    try(PreparedStatement stmt = conn.prepareStatement(sql)) {
      setParameter(stmt, 1, projectId, Integer.class);

      boolean deleted = stmt.executeUpdate() == 1;

      commitTransaction(conn);
      return deleted;
    }
    catch(Exception e) {
      rollbackTransaction(conn);
      throw new DbException(e);
    }
  }
  catch(SQLException e) {
    throw new DbException(e);
  }
}
```