Chapter 6 Cheat Sheet:

- Migrations allow us to modify our application’s data model.
    - Generate user model: `rails generate model <modelName> new` OR `rails generate model <modelName var:varType…>`
    - Migrate up database: `bundle exec rake db:migrate`
    - Roll back database: `bundle exec rake db:rollback`

- Active Record comes with a large number of methods for creating and manipulating data models.
    - Make changes to database: `rails console`
    - “Play” with database (temporary changes): `rails console --sandbox`
        - `User.new[(name: “”, email: “”)]` – create new user [void unless specified], return object
        - `<obj>.valid?` – validate entry, return bool
        - `<obj>.save` – save user to database, return boolean
        - access attribute by name: `<obj>.<attributeName>`
        - `<obj>.create` - create and save entry, return object
        - `<obj>.destroy` destroys object (but still exists in memory), return object
        - `<db>.find(id)` – find user in database, return object or `RecordNotFound`
        - `<db>.find_by(<attr>: “attribute_val”)` – return object
        - `<db>.first` – return first entry
        - `<db>.all` – return all members as ActiveRecord::Relation object (aka an array)
        - Edit a record:
          ```
          <obj>.<attr> = “<newAttrValue”
          <obj>.save
          ```
        - Undo an edit BEFORE SAVING: `<obj>.reload.<attr>`
        - `<obj>.update_attributes()` updates multiple attributes, return boolean
        - `<obj>.update_attribute()` bypasses restrictions for save

- Active Record validations allow us to place constraints on the data in our models.
- Common validations include presence, length, and format.
    - Test the models ONLY: `bundle exec rake test:models`
    - to add presence validation, add the following:
        - add following test to test/models/user_test.rb
          ```
           test "name should be present" do
             @user.name = "     "
             assert_not @user.valid?
          end
          ```
        - add `validates :name, presence: true` to app/models/user.rb

- `<obj>.errors.full_messages` – console call, use to see why you can't save/validate object
- Validate length with `test “<attr> should not be too long”` (Listing 6.14) and set length (Listing 6.16)
  - (user_test.rb) `@user.name = “a” * 51` #string multiplication, 51 char
  - (user.rb) `validates :name,  presence: true,` **`length: { maximum: 50 }`**
- Validate format (like an email) by adding different email styles to user_test.rb (see Listing 6.18 for exact code)
  - Can add a custom error message with

```
assert @user.valid?, "#{valid_address.inspect} should be valid"
```

  - Need to add `VALID_EMAIL_REGEX` to user.rb (Listing 6.21)

• Regular expressions are cryptic but powerful.
  - Set a regular expression to validate the email (table 6.1 in section 6.2.4)

```
VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
```

  - rubular.com is a regular expression editor – has a quick reference guide at bottom and has an error check in it so awesome resource!

• Defining a database index improves lookup efficiency while allowing enforcement of uniqueness at the database level.
  - Add `test "email addresses should be unique"` to user_test.rb and `uniqueness: {case_sensative: false}` to user.rb file
  - Add structure (index) to existing model `rails generate migration add_index_to_users_email`
    - add `add_index :users, :email, unique: true` to new db/migrate/[file]
  - Handle different cases by adding the following to user.rb: `before_save { self.email = email.downcase }`

• We can add a secure password to a model using the built-in `has_secure_password` method.
  - `has_secure_password` – (in user.rb) save hashed *password_digest* to db
    - Need to add *password_digest*:

```
rails generate migration add_password_digest_to_users
password_digest:string
```

    - Need `bcrypt gem` for hash function, add gem and `bundle install`
  - Add `password` and *`password_confirmation`* fields to *@user* in user_test.rb
  - Add `test "password should be present (nonblank)"` and `test "password should have a minimum length"` to user_test.rb (Listing 6.38)
  - `validates :password, presence: true, length: {minimum: 6}` to user.rb to validate user