Black Box Design Techniques

Specification based.

User model

The **User model** is managed by *ActiveRecord* (*Ruby on Rails ORM*) and it defines the core data structure, associations, and validations for *user accounts*, handling both *manual signup* and *OAuth flows* (currently only *Google*), enforcing field formats and lengths, managing relationships, and customizing JSON output to exclude sensitive data like **password_digest** and **email**.

Equivalence Partitions

	Partition Types	Boundary Values	Test case Values
email	invalid	0 - 6 *(string length)*	`'t.d'`,
	valid	6 - 100 *(string length)*	`'johnUser_232@test.dk'`,
	invalid	> 100 *(string length)*	`'a'*200 + '@example.com'`
first_name	valid	2 - 60 *(string length)*	`Maximilian`,
	invalid	> 60 *(string length)*	`'A' * 200`
last_name	valid	2 - 60 *(string length)*	`Wolfeschlegelsteinhausenbergerdorff`,
	invalid	> 60 *(string length)*	`'A' * 200`
bio	valid	0 - 2_000 *(string length)*	`'I like AI food'*15`
	invalid	> 2_000 *(string length)*	`'I like to cook with AI'*500`
uid	valid	0 - 100 *(string length)*	`xoheuif3qr3`

	Partition Types	Boundary Values	Test case Values
	invalid	> 100 *(string length)*	`'u'*450`
provider	valid	0 – 40 *(string length)*	`apple`
	invalid	> 40 *(string length)*	`'a'*100`
password	invalid	0 – 6 *(string length)*	`psw ,`
	valid	6 – 50 *(string length)*	`superSecPsw!a31b_qwe0#`
	invalid	> 50 *(string length)*	`'p'*100`
image_src	invalid	0 – 14 *(string length)*	`http://s.d ,`
	valid	14 – 400 *(string length)*	`https://munchora.pro/uploads/recipes/3r93xhue938383.jpg`
	invalid	> 400 *(string length)*	`'http://img.dk/uploads/' + 'x'*778 (800)`

3-value Boundary Value Analyse

Field	Boundary Condition	Boundary Values	3 Test Case Values (below, on, above)
email	0 - 6 *(string length)*	0	`''`, `'t'`
	6 - 100 *(string length)*	6	`'u@t.d'`, `'u@t.dk'`, `'u1@t.dk'`
	6 - 100 *(string length)*	100	`'a'*87 + '@example.com'` (99), `'a'*88 + '@example.com'` (100), `'a'@89 + '@example.com'` (101)
first_name	0 - 2	0	`''`, `0`
	2 - 60 *(string length)*	2	`0`, `Li`, `'Lee'`
	2 - 60 *(string length)*	60	`'A'*59`, `'A'*60`, `'A'*61`
last_name	0 - 2	0	`''`, `J`
	2 - 60 *(string length)*	2	`J`, `Jo`, `'Joe'`
	2 - 60 *(string length)*	60	`'A'*59`, `'A'*60`, `'A'*61`
bio	0 - 2_000 *(string length)*	0	`''`, `'a'`
	0 - 2_000 *(string length)*	2000	`'a'*1_999`, `'a'*2_000`, `'a'*2_001`
uid	0 - 100 *(string length)*	0	`''`, `'a'`
	0 - 100 *(string length)*	100	`'a'*99`, `'a'*100`, `'a'*101`
provider	0 - 40 *(string length)*	0	`''`, `'u'`

Field	Boundary Condition	Boundary Values	3 Test Case Values (below, on, above)
	0 - 40 *(string length)*	40	`'u'*39`, `'u'*40`, `'u'*41`
password	0 - 6 *(string length)*	0	`''`, `'p'``
	6 - 50 *(string length)*	6	`'secpw'`, `'secpsw'`, `'secpswd'`
	6 - 50 *(string length)*	50	`'p'*49`, `'p'*50`, `'p'*51`
image_src	0 - 14 *(string length)*	0	`''`, `'h'`
	14 - 400 *(string length)*	14	`'http://img.dk'` (13), `'http://img.dk/'` (14), `'http://img.dk/1'` (15)
	14 - 400 *(string length)*	400	`'http://img.dk/uploads/' + 'x'*377` (399), `'http://img.dk/uploads/' + 'x'*378` (400), `'http://img.dk/uploads/' + 'x'*379` (401)

Edge Cases

This outlines the field-level test cases for the **User model**, specifying expected formats and example invalid values to ensure robust validation against **incorrect types**, **malformed emails**, and **invalid URLs**.

Field	Format	Test Case Value
email	`URI – :mailto`	`null`, `plainaddress`, `@missinguser.com`, `user@.com`, `user@com,com`, `user@exa mple.com`
first_name	`UTF–8 encoded Unicode string`	`wrong data types`
last_name	`UTF–8 encoded Unicode string`	`wrong data types`
bio	`UTF–8 encoded Unicode string`	`wrong data types`
uid	`UTF–8 encoded Unicode string`	`wrong data types`
provider	`UTF–8 encoded Unicode string`	`wrong data types`
password	`UTF–8 encoded Unicode string`	`wrong data types`
image_src	`URI – :http/https`	`a.com`, `invalid_url`, `http://a.co`, `htp://site.com/uploads/x.jpg`

Decision table

This table summarizes the key conditions and expected outcomes for **user creation**, covering both *manual signup* and *OAuth flows*, and highlights which combinations of field validity result in a successful account creation.

User model	R1	R2	R3	R4	R5	R6	R7	R8
Conditions								
Email & names valid?	F	T	F	T	F	T	F	T
Password valid?	F	F	T	T	/	/	/	—
Provider & uid present?	/	/	/	/	F	F	T	T
Image_src valid?	/	/	/	/	/	/	/	/
Bio valid?	/	/	/	/	/	/	/	/
Expected Validation Context	Manual signup	Manual signup	Manual signup	Manual signup	OAuth signup	OAuth signup	OAuth signup	OAuth signup
Actions								
User created (valid)?	F	T	F	T	F	F	F	T